PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 15/60, 9/44 — A1

(11) International Publication Number: WO 95/32476

(43) International Publication Date: 30 November 1995 (30.11.95)

(21) International Application Number: PCT/US95/06791

(22) International Filing Date: 25 May 1995 (25.05.95)

(30) Priority Data:
08/248,752 — 25 May 1994 (25.05.94) — US

(71) Applicant: CADENCE DESIGN SYSTEMS, INC. [US/US]; 555 River Oak Parkway, San Jose, CA 95134 (US).

(72) Inventors: BANKS, Steven, Aaron; 794 Belfair Court, Sunnyvale, CA 94087 (US). CARILLI, Marco; Via Pierfrance Bonetti, 13, I-00128 Rome (IT). WONG, Franklin, Lee; 2268 Almaden Road, San Jose, CA 95125 (US).

(74) Agents: McNELIS, John, T. et al.; Fenwick & West, Suite 600, Two Palo Alto Square, Palo Alto, CA 94306 (US).

(81) Designated States: AM, AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU, LV, MD, MG, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TT, UA, UG, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), ARIPO patent (KE, MW, SD, SZ, UG).

Published

With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: A SYSTEM AND METHOD FOR CREATING DESIGN CONFIGURATIONS AND FOR CONTROLLING THE EXECUTION OF MULTIPLE DESIGN TOOLS

(57) Abstract

A system and method are described for controlling the execution of design tools and for creating design configurations. The system comprises a flow sequencer, a flow map, and a flow configuration object. The flow sequencer controls the execution of design tools in accordance with the flow map and constructs a configuration of its actions in the active flow configuration object. The flow sequencer moves through a flow map executing the actions required by each node of the flow map. At certain nodes the flow sequencer executes design tools. At other nodes the flow sequencer gathers and manipulates data so that a design tool can utilize the data. The configuration is constructed as the nodes of the flow map are executed. The flow sequencer is also capable of analyzing a design to determine whether the data associated with each node is the most current data available. The flow sequencer can identify the nodes where the data is not most current and can re-execute the nodes necessary to update the design. The flow sequencer will execute the minimum number of nodes necessary to update the design. The system also includes a means for debugging a design. At any node the system can stop the execution of the nodes of the flow map and output the current state of the design. The system will then continue executing the nodes from the node at which it stopped the execution of the nodes.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgystan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo			SE	Sweden
CH	Switzerland	KR	Republic of Korea	SI	Slovenia
CI	Côte d'Ivoire	KZ	Kazakhstan	SK	Slovakia
CM	Cameroon	LI	Liechtenstein	SN	Senegal
CN	China	LK	Sri Lanka	TD	Chad
CS	Czechoslovakia	LU	Luxembourg	TG	Togo
CZ	Czech Republic	LV	Latvia	TJ	Tajikistan
DE	Germany	MC	Monaco	TT	Trinidad and Tobago
DK	Denmark	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	US	United States of America
FI	Finland	ML	Mali	UZ	Uzbekistan
FR	France	MN	Mongolia	VN	Viet Nam
GA	Gabon				

A SYSTEM AND METHOD FOR CREATING DESIGN CONFIGURATIONS AND FOR CONTROLLING THE EXECUTION OF MULTIPLE DESIGN TOOLS

Background Of The Invention

1. Field of the Invention

This invention relates generally to computer aided design systems, and in particular, to a system and method for controlling the execution of design tools in an electronic design system.

2. Description of the Related Art

As electronic systems and devices have become more complex and difficult to design, designers have used computer and electronic tools to deal with the increased complexity. These computer and electronic tools are referred to as electronic design automation ("EDA") and have become the primary tools used by designers to design new electronic devices. The number of EDA tools has increased so dramatically that the coordination and control of different EDA tools is now a major concern.

Several significant problems presently confront EDA. The sequencing, coordination, and efficient use of design tools is one of these problems. EDA tools are cumbersome and difficult to use. Many tools require numerous data files as inputs and produce multiple output files. When a designer executes a series of tools, the coordination and control of the inputs and outputs of each tool is quite difficult. These difficulties are magnified when several tools are executed in parallel. Often, different EDA tools are created and manufactured by different companies. The inputs and outputs of the one EDA tool may not be compatible with other EDA tools. Modifying the output of one EDA tool so that it may be used as an input to a different EDA tool is, in many cases, difficult and cumbersome.

A design process is the use of a sequence of EDA tools to move from a set of design specifications to a completed design. Many EDA tools must be executed in a fixed order. The sequence in which tools are to be executed is called a design methodology. Design methodologies must be strictly enforced. Failure to execute a tool in the proper sequence can invalidate the output of that tool, and thus, an entire design. Moreover, correcting the

improper execution of tools can be quite costly. Thus, a system and method of enforcing design methodologies is needed to eliminate costly failures to follow design methodologies.

Often designers modify designs only slightly from an earlier version of the design. An incremental modification of one portion of a design process, however, may require the running of many design tools and the concomitant problems of coordination and control of the design tools. Moreover, the designer may be interested in only a portion of the design process. Therefore, the designer may not need to re-execute the entire design process. Frequently with current EDA tools, incremental changes require re-executing the entire design process to ensure accuracy. This is also problematic because of the time required to re-execute all the tools in the design process. Thus, there is a need for a system and method for evaluating a design process and re-executing only the portion of the design process affected by the incremental changes.

To address these problems in EDA, computer systems have been developed to complement EDA by enforcing design methodologies and by providing a means to analyze the design process. These prior art systems have adopted a variety of approaches for addressing the problems of EDA. They analyze the design process after execution to ensure compliance with the design methodology. These prior art systems rely on a table of rules, a chronology of the creation of data files and a chronology of the execution of design tools to determine the steps in the design process. These systems are heavily dependent upon a chronological analysis of the data files to generate the dependencies of each step in the design methodology. These systems are unable to construct an actual design history, and can only infer a very rudimentary history after the design process has been run. They are, therefore, susceptible to inaccuracies and cannot be used to recreate a design.

Other systems create a history of the design process by inference and allow the user to analyze stages of the design process. These systems rely on a table of rules and the design environment to infer the steps in the design process and to enforce the design methodology. The histories are inferred and are reconstructed after the design process has been run. They are severely limited in the data they contain and in the accuracy of that data. These systems are not capable of producing an exact, complete history of the design process. Furthermore,

existing systems do not aid the designer in coordinating the design tools, the design tool inputs, or the design tool outputs.

The prior art approaches analyze the design processes as a whole. They only infer and reconstruct the design history after the design has been run. They also do not enforce the design methodology as the design tools are being executed. Thus, there is a need for a system and method for controlling the execution of multiple design tools that provides efficient and effective enforcement of design methodologies and for simultaneously creating design configurations of the design process.

Summary Of The Invention

The present invention overcomes the limitations and shortcomings of the prior art with a system and method for enforcing design methodologies and analyzing design processes. The system of the present invention includes a display device, a processor, an input device, a memory means, and a data storage device. The components are preferably coupled together by a bus. The memory advantageously comprises a flow sequencer, an active flow configuration object memory ("active FCO memory"), and a flow map memory. The flow sequencer is a set of program instruction steps, which the system of the present invention uses to execute a specific design process described by the flow map. The flow sequencer moves through the flow map, stored in the flow map memory, from node to node. The flow map is a pattern of nodes; each node requires an action by the flow sequencer. The flow sequencer establishes the dependencies for each node in the flow map. At tool nodes, the flow sequencer calls for the execution of a design tool, stores the data produced by the design tool, and stores, in the active FCO, a record of the action taken, the data produced, and the design environment. At data nodes the flow sequencer gathers data produced by earlier executed tool nodes and labels, formats, and otherwise modifies such data so that a succeeding tool node can use the data in its design tool. In this way, the present invention allows the use of otherwise incompatible design tools without modifying the design tools themselves. Once the FCO has been generated by the flow sequencer, it can then be used by the flow sequencer to

re-execute quickly portions of the design, or for analysis of the outputs of any tool used in the design process.

The present invention also comprises a method for executing design tools and constructing a history of the design process. The preferred method for executing design tools begins by marking or identifying those nodes of the flow map that the system will execute. The system marks the nodes that the user indicates and those nodes that must be executed due to dependencies upon the indicated nodes. The system next executes design tools at tool nodes. The system modifies and prepares data for succeeding design tools at data nodes. The process of executing the design tools in the proper order to create a design is referred to as a design flow run. The method of executing the design tools and constructing a history of the design flow run preferably comprises the steps of: analyzing the nodes upon which the current node depends; executing the appropriate design tool if the dependencies are valid; storing a record of the design tool executed; storing a record of the surrounding design environment; storing a reference to the output of the design tool in the active FCO at the next data node; and marking the nodes as analyzed. This process is repeated for all marked nodes. In this way, the flow sequencer executes the design methodology described by the flow map and builds a complete history of the design process in the active FCO. The system completes a design flow run when all nodes that had been marked for execution have been executed.

The present invention also includes a method for updating a design flow run. The method for updating a design flow run is referred to herein as "design make." The design make function searches for nodes of the flow map that have been modified. The system then re-executes only those nodes that have been modified or that depend upon such modified nodes. In this manner, the system updates a design process without re-executing all nodes of the flow map.

The system also includes a method for debugging a design process. The user may specify any number of nodes as break nodes to aid in the debugging of the design process. The system will stop at break nodes, output the data from the immediately preceding node, and wait for a command to continue. This method allows the user to review the data at any point of the design process, during the design process.

Brief Description Of The Drawings

Figure 1 is a block diagram of a preferred embodiment of a system for creating design configurations and for controlling the execution of multiple design tools;

Figure 2 is a block diagram of the preferred embodiment for a data storage device in the system of the present invention;

Figure 3 is a block diagram of the preferred embodiment for a memory in the system of the present invention;

Figure 4 is a block diagram showing data paths between components of the system of the present invention;

Figure 5 is a flowchart showing the preferred method for executing design tools and creating design configurations specified by a design methodology with the present invention;

Figures 6A & 6B are flowcharts showing the preferred method for identifying or marking the portion of a design methodology that is to be executed;

Figure 7 is a flowchart showing the preferred method for initializing the execution of a design methodology;

Figure 8 is a flowchart showing the preferred method for selecting a node of a design methodology that is to be executed;

Figure 9 is a flowchart showing the preferred method for executing a node of a design process according to the present invention;

Figures lOA and lOB are flowcharts showing the preferred method for re-executing tools specified by design make with the system of the present invention; and

Figure 11 is a flowchart showing the preferred method for debugging a design process according to the present invention.

Description Of The Preferred Embodiment

Referring now to Figure 1, a block diagram of the preferred embodiment of a system 11 is shown. Data is entered into the system 11 for creating design configurations and for controlling the execution of multiple design tools. Among the data input is a flow map. The

flow map embodies a design methodology and specifies the design tools that must be executed and the data that must be stored during a design process. The system 11 implements the design process by executing the design tools and storing data in the order indicated by the flow map. In this way, system 11 enforces the design methodology. As the system 11 is executing the design process, it constructs a complete history of the design process. At each step in the design methodology, the system 11 records the action taken, the outputs of the action, and the current design environment.

The system 11 can also update a design process. The system 11 automatically searches through the design process comparing the data most recently created with the data stored in the history of the design process. The system 11 determines where the most current data does not match the data recorded in the history. The system 11 then re-executes those portions of the flow map necessary in order to update the design. In this way, the system 11 efficiently updates a design. The system 11 does not re-execute the entire design process, just those portions necessary to update the design.

The system 11 preferably comprises an input device 10, a display device 16, a processor 18, a data storage device 22, and a memory 24. The input device 10, the display device 16, the processor 18, the data storage device 22, and the memory 24, are preferably coupled together by a bus 34 in a Von Neuman architecture. Those skilled in the art will realize that these components 10, 16, 18, 22, 24, and 34 may be coupled together according to a variety of other computer architectures without departing from the spirit or scope of the present invention. The system 11 is preferably a workstation from Sun Microsystems Computer Corporation of Palo Alto, California.

The input device 10 is a means for inputting data and commands from a user to the system 11. The preferred embodiment of the input device 10 is a keyboard and mouse type controller.

The display device 16 is preferably a monitor used for displaying information to the user. The display device 16 may be a conventional CRT type display device.

The processor 18 executes program instruction steps, generates commands, and analyzes data configurations according to program instruction steps that are stored in the

memory 24 and in the data storage device 22. The processor 18 preferably is a microprocessor such as the Sun Sparc from Sun Microsystems Computer Corporation of Palo Alto, California.

Referring now to Figure 2, the data storage device 22 is shown in more detail. The data storage device 22 is preferably a hard disk drive. The data storage device 22 includes a design tool storage 38, a flow map storage 42, a run configuration object storage 40, and a flow configuration object storage 36. The design tool storage 38, the flow map storage 42, the run configuration object storage 40, and the flow configuration object storage 36 are coupled to the bus 34. While the flow configuration object storage 36, the design tool storage 38, the run configuration object storage 40, and the flow map storage 42 will now be described as separate storage devices, those skilled in the art will realized that they may be portions or sections of a single hard disk drive.

The design tool storage 38 is preferably a data storage device for storing various design tools. The design tools are stored in the design tool storage 38 when they are not being used. The design tool storage 38 stores and retrieves design tools in response to commands from the processor 18. In an exemplary embodiment, the design tools stored in the design tool storage include design rule checkers, verifiers, work benches, placers, and synthesizers. The system 11 may be used with any tools, tasks, or processes to implement any design methodology.

The flow map storage 42 is a means for storing flow maps when such flow maps are not in use by the system 11. A flow map is a series of design steps that must be executed during a design process. The flow map also indicates the order in which the steps must be executed. The flow map is made up of a series of nodes. There are four types of nodes: tool nodes, data nodes, control nodes, and sub-flow nodes. A tool node indicates the execution of a design tool. A data node indicates the manipulation of data. A data node need not immediately follow a tool node, but in the preferred embodiment, tool nodes and data nodes are paired with a data node immediately following a tool node to manipulate and to arrange the outputs of the tool node. A tool node may receive input from one or more nodes, but in the preferred embodiment, a data node receives one input from the immediately preceding tool

node. A control node is a node that allows for the modification of the status of other nodes within the flow map. A sub-flow node is a node that contains, within itself, a design methodology. When the sub-flow node is executed, a complete design flow run, which the sub-flow node represents, is executed.

Each node has two main indicators of its status. Other indicators of status may be used. A first indicator for a node is active or inactive. If a node is marked active, this means that the node is to be executed during the design flow run. When only a portion of the flow map is to be executed during the design flow run, the nodes that are not to be executed are marked inactive. When an active node is executed, it is also marked inactive. An inactive node either has been executed during the design flow run or is not to be executed during the design flow run. A second indicator for each node is evaluated or unevaluated. An evaluated node has data associated with it; an unevaluated node does not have data associated with it. In order for a node to be executed, all nodes upon which such node depends must be evaluated and inactive. The fact that a node is marked evaluated does not guarantee that the data associated with that node is valid, rather only that the data associated with that node exists.

The nodes are linked together by connectors that indicate the dependencies of a particular node upon other nodes. These connectors are references within each node that identify the nodes that input to such node and the nodes to which such node directs its outputs. A node is said to be dependent upon all nodes that input to such node and upon all nodes that the inputting nodes are in turn dependent. Thus, a node is dependent upon all nodes "up stream" from such node or from which it directly or indirectly receives inputs. The initial nodes of the flow map are referred to as input nodes. The input nodes may require that the user supply data. The output nodes of the flow map are referred to as terminal nodes. The terminal nodes are generally the final nodes of the flow map. The terminal nodes preferably are data nodes or control nodes and not tool nodes, or sub-flow nodes. In some situations it is particularly advantageous to use control nodes as the terminal nodes of the flow map. The function of a control node may be to make a decision whether or not to terminate the design flow run. In order to limit the nodes that will be executed and focus on a particular portion of a design, the user may designate nodes that are within the flow map as input nodes or as

terminal nodes. All nodes up stream from the nodes labeled as input nodes will be ignored, and the system 11 will look to the user for inputs at the nodes labeled as input nodes. Likewise, the system 11 will ignore all nodes below the nodes labeled as terminal nodes. The flow map storage 42 stores and retrieves flow maps in response to commands from the processor 18.

The run configuration object storage 40 is preferably a data storage device for storing run configuration objects ("RCOs") when such RCOs are not in use by the system 11. An RCO is a record of the inputs to a design tool, the outputs of the design tool, the identity of the design tool that produced the data, a record of the active configuration at the time the design tool was executed, a record of the time at which the design tool was executed, and a record of the tool node from which the design tool was executed. The RCO contains a reference to its version as set by the user. The RCO may also contain a record of the status of the execution of the design tool such as a UNIX status of success or failure of the execution of the design tool. A separate RCO is created each time a design tool is executed. The run configuration object storage 40 stores and retrieves RCOs in response to commands from the processor 18.

The flow configuration object storage 36 is a means for storing flow configuration objects ("FCOs") when such FCOs are not being used by the processor 18 and the memory 24. An FCO is a detailed history of a design flow run. The history includes the locations in the run configuration object storage 40 of the RCOs that were created during the design flow run, the versions of the RCOs, the active configuration that existed when each design tool was executed, and other data. The data is arranged in the order in which the nodes of the flow map were executed. At the end of a design flow run, a record of the time the design flow run was completed is stored in the time stamp of the FCO. The use of the FCO is particularly advantageous because it is constructed during the design flow run. An FCO is not an inferred history or a history that was some how derived after the design process is completed. The FCO is an exact record which may be used to re-create a design process, used as a basis for modifying a design process, used as a tool for detecting flaws in the design process, used as a basis for creating other design processes, and used as a basis for establishing experimental models. The FCO becomes a tool for creating and perfecting design processes. The flow

configuration object storage 36 stores and retrieves FCOs in response to commands from the processor 18.

Referring now to Figure 3, a block diagram of the memory 24 is shown. Referring now also to Figure 4, a block diagram is shown that illustrates an overview of the flow of data in the system 11 of the present invention. The memory 24 is preferably a dynamic random access memory. The memory 24 stores data, operating systems, and program instruction steps. The memory 24 comprises an active configuration memory 32, a flow map memory 48, an active FCO memory 66, an active RCO memory 50, a flow sequencer 54, a flow file description memory 12, a flow file reader 26, a flow generator 28, a flow viewer 56, a flow editor 52, an active tool memory 60, a tool notification memory 44, a tool launch server 64, and a graphical user interface memory 62.

The active configuration memory 32 is a record of the current, surrounding environment and default conditions of the system 11. The active configuration memory 32 may record tolerances for outputs and design assumptions. For example, the active configuration memory 32 may contain search lists, set up files, data on how design tools will be executed, versions of input and design data files, and status information. The active memory 32 also may contain reference to data files which may be, or may have been, modified by a design tool. The processor 18 regularly updates the active configuration memory 32. The active configuration 32 is coupled to the bus 34.

The flow map memory 48 is a means for storing the flow map that the system 11 is currently executing. The flow map memory 48 is coupled to the bus 34.

The active FCO memory 66 is a means for storing the FCO that the system 11 is currently using. The active FCO memory 66 is coupled to the bus 34.

The active RCO memory 50 is a means for storing the RCO that the system 11 is currently using. Design tools may store their output data in an RCO. When the flow sequencer 54 executes a tool node of a flow map, the flow sequencer 54 creates a new RCO which it stores in the active RCO memory 50. The flow sequencer 54 copies the input data to the design tool into the RCO and hands this RCO and the design tool that is to be executed to another system for execution of the design tool. When the design tool has completed

execution, it records its outputs in the new RCO and returns the new RCO to the active RCO memory 50. The RCO is then stored in the run configuration object storage 40. An RCO may also be stored in the active RCO memory 50 when the flow sequencer 54 is determining whether or not the tool node that created the data of the RCO must be re-executed in order to update a design. The RCO stored in the active RCO memory 50 is, therefore, the RCO that the flow sequencer 54 has just created in anticipation of the execution of a design tool, the RCO that contains the output data of a design tool that was just executed, or an RCO that is being analyzed by the flow sequencer 54 for the validity of its data. The active RCO memory 50 is coupled to the bus 34.

The flow sequencer 54 is a set of program instruction steps that are stored in the memory 24. The processor 18, under the control of the program instruction steps of the flow sequencer 54, executes the design process described by the flow map stored in the flow map memory 48 and constructs a history of the design process in the active FCO memory 66. When the flow sequencer 54 is said to take an action, it is actually the processor 18 under the control of the program instruction steps of the flow sequencer 54 taking the action. The flow sequencer 54 steps forwards or backwards from node to node through the flow map calling for the execution of design tools at tool nodes, the storing of data at data nodes, executing the functions of control nodes, and executing the design said to take an action, it is actually the processor 18 under the control of the program instruction steps of the flow sequencer 54 taking the action. The flow sequencer 54 steps forwards or backwards from node to node through the flow map calling for the execution of design tools at tool nodes, the storing of data at data nodes, executing the functions of control nodes, and executing the design flow runs of sub-flow nodes. When the flow sequencer 54 executes a tool node, the flow sequencer 54 retrieves RCOs from the run configuration object storage 40 as indicated by the nodes that input to the tool node, retrieves the appropriate design tool from the design tool storage 38, and prepares a new RCO for the execution of the design tool. The flow sequencer 54 stores the new RCO in the active RCO memory 50. In the preferred embodiment design tools are executed by a separate system (not shown). The separate system is a server that supports the running and monitoring of individual design tools. The present invention arranges data so that

each design tool need not be modified to work with other design tools. In this way, the present invention allows the use, in combination, of design tools from many different manufacturers which otherwise would be incompatible for such use in combination. The flow sequencer 54 stores copies of the input data files in the RCO and presents the design tool and the new RCO to the other system for the execution of the design tool. When the other system has completed execution of the design tool, the RCO is returned to the active RCO memory 50. The flow sequencer 54 then stores the RCO in the run configuration object storage 40 and creates an entry in the FCO stored in the active FCO memory 66. The entry in the FCO contains the location of the RCO in the run configuration object storage 40 and a record of the active configuration, as stored in the active configuration memory 60 at the completion of the execution of the design tool. When the flow sequencer 54 executes a data node, the flow sequencer 54 gathers data files. These data files may be stored in RCOs, may be data so that it is formatted correctly for, or is otherwise acceptable to, a succeeding tool node. In this way, the present invention allows for use of different design tools, without modification of the design tools, which otherwise would be incompatible for use in one design flow run. At a control node, the flow sequencer 54 executes a function specified by the control node. The flow sequencer 54 analyzes data, information stored in the active configuration memory 32, and the general design environment to determine the result of the function specified by the control node. The flow sequencer 54 may, in response to the results of the function of the control node, mark certain nodes of the flow map active or inactive. In general, at a control node the flow sequencer 54 may modify, according to the function specified by the control node, the execution of the design flow run in light of the results of the execution of the design flow run up to the control node. At sub-flow nodes, the flow sequencer 54 executes the design flow run specified by the sub-flow node. The design flow run specified by the sub-flow node is a complete design flow run; during execution of this design flow run, the flow sequencer 54 will create a new and separate flow configuration object. Note that any design flow run may be specified by a control node within a design flow run. The output of a sub-flow node is an FCO.

The flow sequencer 54 also controls the design updating function. The flow sequencer 54 compares the data associated with each node to the data associated with the surrounding nodes. The flow sequencer 54 determines when the data is out of date and determines the nodes that must be re-executed to update the design. The flow sequencer 54 then re-executes only the nodes necessary to update the design. The flow sequencer 54 is coupled to the bus 34. In particular, the flow sequencer 54 is coupled, as shown in Figure 4, to receive a flow map, to access and to receive an RCO, and to access and to receive an FCO.

The flow file description memory 12 is preferably a means for storing a flow file description. The flow file description is an ASCII description of a flow map. The user inputs the flow file description through the input device 10. The flow file description memory 12 is coupled to the bus 34.

The flow file reader 26 is a set of program instruction steps that are stored in the memory 24. When the processor 18 executes these program instruction steps, the processor 18 reads and parses the data of the flow file description, which is stored in the flow file description memory 12, in preparation for generating a flow map. When the flow file reader 26 is said to execute an action, in actuality, the processor 18 is executing the program instruction steps of the flow file reader 26. The flow file reader 26 is coupled to the bus 34 and, in particular, is coupled, as shown in Figure 4, to receive a flow file description.

The flow generator 28 is a set of program instruction steps that are stored in the memory 24. When executed by the processor 18, the flow generator 28 accepts an input from the flow file reader 26 and generates a flow map that the flow sequencer 54 can execute. The flow generator 28 receives the parsed data of the flow file description from the flow file reader 26. When the flow generator 28 finds enough information to form a node of the flow map, the flow generator 28 establishes a node and establishes the dependencies of the node. The flow generator 28 stores the complete flow map in the flow map memory 48. When the flow generator 28 is said to execute an action, in actuality, the processor 18 is executing the program instruction steps of the flow generator 28. The flow map generator 28 is coupled to the bus 34. As Figure 4 shows, the flow generator 28 is coupled to receive data from the flow file reader 26 and to output a flow map.

The flow viewer 56 is a set of program instruction steps that are stored in the memory 24. When the processor 18 executes these program instruction steps, the flow viewer 56 is a means of outputting, through the display device 16, data concerning the flow map that is stored in the flow map memory 48, the flow sequencer 54, and the FCO that is stored in the active FCO memory 66. The flow viewer 56 is also a means by which the user, through the input device 10, may modify the parameters of a design flow run. The user may modify the FCO that is stored in the active FCO memory 66 or the active configuration that is stored in the active configuration memory 32. The user may also temporarily modify the flow map that is stored in the flow map memory 48 using the flow viewer 56. The data that the user enters through the input device 10 and the flow viewer 56 affects only the current design flow run. The user may not permanently modify a flow map, the flow sequencer 54, or an FCO through the flow viewer 56. The flow viewer 56 is coupled to the bus 34. In particular, the flow viewer 56 is coupled, as shown in Figure 4, to transmit data to and to receive data from the flow sequencer 54, the flow configuration object, and the flow map.

The flow editor 52 is a set of program instruction steps that are stored in the memory 24. When the processor 18 executes these program instruction steps, they are a means by which a user can, through the display device 16 and the input device 10, view and modify the flow file description that is stored in the flow file description memory 12, the flow map that is stored in the flow map memory 48, the flow sequencer 54, or the FCO that is stored in the active FCO memory 66. Using the flow editor 52, the user can create a completely new design process. The user may permanently modify the flow file description stored in the flow file description memory 12, the flow map stored in the flow map memory 48, or the FCO by means of the flow editor 52. The flow editor 52 is coupled to the bus 34. As is shown in Figure 4, the flow editor 52 is coupled to transmit data to and to receive data from the flow configuration object, the flow map, and the flow sequencer 54. The flow editor 52 is also coupled to transmit data to the flow file description.

The active tool memory 60 is a reference to the design tool that is currently being executed. The execution of a design tool is called a tool run. The active tool memory 60 is coupled to the bus 34.

The system 11 also includes a design tool completion signal called a tool notification. Tool notifications are stored in a tool notification memory 44. This signal informs the flow sequencer 54 of the completion of the execution of a design tool and of the tool node from which the flow sequencer 54 called the design tool. The tool notification memory 44 is coupled to the bus 34.

The tool launch server 64 is a set of program instruction steps stored in the memory 24 that, when executed by the processor 18, serve as a conduit for data and commands from the processor 18 and the flow sequencer 54 to another system for the execution of the design tool. The tool launch server 64 hands off the active tool, input data, and the RCO that is stored in the active RCO memory 50 to a separate system to execute the design tool. The tool launch server 64 is coupled to the bus 34.

The graphical user interface memory 62 is a means for storing a plurality of graphical user interfaces. The graphical user interfaces are screen formats, fields, and icons that are displayed to the user on the display device 16. The graphical user interfaces are used in conjunction with the flow editor 52 and the flow viewer 56. The graphical user interface memory 62 is coupled to the bus 34.

Referring now to Figure 5, a preferred method for creating design configurations and for controlling the execution of multiple design tools is shown. Figure 5 shows the preferred method for executing a design flow run. During a design flow run, the system 11 implements the design process described in the flow map that is stored in the flow map memory 48. During the design flow run, the system 11 advantageously constructs a history of the design flow run that is a complete design configuration. Beginning in step 100, the system 11 accepts a flow map and a set of initial conditions as inputs. The flow map is loaded into the flow map memory 48. The user may instruct the system 11 to execute all nodes of the flow map, or the user may limit the design flow run to a selected portion of the flow map. If the user limits the design flow run to a portion of the flow map, the user will also specify in the initial conditions the portion of the flow map that is to be executed during the design flow run. The system 11 will treat the initial nodes of such portion of the flow map as input nodes and the final nodes of such portion of the flow map as terminal nodes. In step 102, the system 11

identifies or marks the nodes of the flow map as active. The details of how the system 11 marks nodes active is explained below with reference to Figures 6A & 6B. The system 11 initializes a design flow run in step 104. The details of an initialization of a design flow run are shown in Figure 7. In step 106, the system 11 selects an active node to be executed. The system 11 then executes the selected node in step 108. In step 110, the system 11 searches for other active nodes in the flow map. If there are active nodes, the system 11 returns to step 106 to select and to execute another active node. If in step 110 there are no active nodes on the flow map, the design flow run is complete.

Referring now to Figures 6A & 6B, the preferred method for identifying or marking nodes as active is shown. When a node is marked, the appropriate signal in the flow map memory 48 is set or reset. The flow sequencer 54 first, in step 112, marks all nodes of the flow map as inactive. The flow sequencer 54 then in step 114 analyzes the initial conditions from the user to determine if the entire flow map will be executed or if only a portion of the flow map will be executed. If the entire flow map will be executed, the flow sequencer 54 marks all nodes active in step 116, then marks all nodes unevaluated in step 118 and the method ends.

If only a portion of the flow map will be executed, the flow sequencer 54 marks the pertinent nodes active. The pertinent nodes are those nodes that make up the portion of the flow map that is to be executed. The preferred method for marking nodes proceeds from step 114 to step 120 where the user inputs one or more nodes of interest and designates the nodes as either input nodes or output nodes. This data may be a part of the initial conditions originally input to the system 11. The nodes of interest should be similarly categorized as input nodes or output nodes. The flow sequencer 54 tests whether the nodes of interest are input nodes in step 124. If the nodes of interest are designated input nodes, the flow sequencer 54 proceeds to step 126 where it marks the nodes listed as input nodes. The flow sequencer 54 then goes to an inactive input node in step 128. At step 130, the flow sequencer 54 marks the node active. In the preferred method, the flow sequencer 54 then proceeds to step 132 where the flow sequencer 54 analyzes the current node for its status as a terminal node. If the node is not a terminal node, the flow sequencer 54 proceeds to step 136 where the

flow sequencer 54 determines if all nodes to which the current node directs its outputs are active. If all nodes are active, the flow sequencer 54 has completed this branch of the flow map, and the flow sequencer 54 proceeds to step 144. If there are inactive nodes to which the current node directs its outputs, the flow sequencer 54, in step 134, selects the next inactive node in the flow map as the current node. With the newly selected node, the flow sequencer 54 returns to step 130.

If the current node in step 132 is found to be a terminal node, the flow sequencer 54 proceeds to step 138 where the flow sequencer 54 determines whether there are inactive nodes to which the current node directs its outputs. The first time this determination is made the result will be negative since the flow sequencer 54 is at a terminal node. The flow sequencer 54 will, however, be moving up this branch of the flow map searching for branches that originate off of this branch. As the flow sequencer 54 moves up the branch, the test in step 138 will become significant because the flow sequencer 54 may find branches of the flow map, which originate from the current branch, that have not yet been marked as active. If there are inactive nodes to which the current node directs its outputs, the flow sequencer 54 returns to step 134 to select such node and to mark the nodes down this branch as active.

If in step 138 there are no inactive nodes to which the current node directs its outputs, the flow sequencer 54 determines if the node is an input node in step 144. The flow sequencer 54 may reach step 144 either from step 138 or from step 136. If the node is not an input node, the flow sequencer 54, in step 142, moves backward through the flow map to an active node that directs its outputs to the current node. The flow sequencer 54 then returns to step 138. If in step 144 the node is an input node, the flow sequencer 54 has marked all nodes that depend upon this input node active. The flow sequencer 54 then looks for inactive input nodes in step 146. If there are inactive input nodes, the flow sequencer 54 returns to step 128 to mark such inactive input node, and all nodes that depend on such input node, active. If there are no inactive input nodes, the flow sequencer 54 marks all active nodes as unevaluated in step 148, and the method ends.

If in step 124 of Figure 6A the nodes of interest are not input nodes, the nodes of interest are output nodes, and the flow sequencer 54 proceeds to step 150 of Figure 6B. The

flow sequencer 54 proceeds from step 124 to step 150 where it identifies the nodes as terminal nodes. The flow sequencer 54 then selects an inactive terminal node in step 152. The flow sequencer 54 marks the current node active in step 154. The flow sequencer 54 then searches backwards through the flow map for inactive nodes that direct their outputs to the current node, step 158. If there are inactive nodes that direct their outputs to the current node, the flow sequencer 54, in step 156, selects such an inactive node as the current node and returns to step 154.

If in step 158 there are no inactive nodes that direct their outputs to the current node, the flow sequencer 54 begins moving forward through the flow map looking for incoming branches. The flow sequencer 54 determines if the current node is a terminal node in step 160. If the node is not a terminal node, the flow sequencer 54 selects the next active node to which the current node directs its outputs in step 162 and returns to step 158. In this way the flow sequencer 54 retraces its steps to the terminal node from which it began in step 152. If in step 160 the node is a terminal node, the flow sequencer 54 has returned to the initial terminal node and has marked all nodes upon which this terminal node depends as active. The flow sequencer 54 then searches the flow map for inactive terminal nodes in step 164. If there are inactive terminal nodes, the flow sequencer 54 returns to step 152 and selects an inactive terminal node as the current node. If in step 164 there are no inactive terminal nodes, the flow sequencer 54 marks all active nodes as unevaluated in step 166, and the method ends.

An alternate method eliminates steps 124 through 148. When the nodes of interest are input nodes, steps 124 through 148 require the flow sequencer 54 to make two passes through the nodes. The first pass is embodied in steps 124 through 148; the second pass is made when the nodes are executed. The alternate method involves eliminating the first pass. In this alternate method, when the nodes of interest are input nodes, the flow sequencer 54 executes a node; the flow sequencer 54 then executes all nodes to which the just executed node directs its outputs. This process is repeated until the terminal nodes are reached. In general, the flow sequencer 54 executes a node and then executes the nodes to which the just executed node directs its outputs. While this alternate method eliminates the need to mark the appropriate nodes active, it limits the opportunities to execute nodes in parallel. For some flow maps,

however, the alternate method may be faster and more efficient than the preferred method shown in steps 122 through 148.

Referring now to Figure 7, the preferred method for initializing a design flow run (step 104 of Figure 5) is shown. Beginning in Step 170, the flow sequencer 54 selects an active input node, and in step 172, determines whether all the input data for the selected node is available. The flow sequencer 54 may make this determination by analyzing a file system or a design management system. The flow sequencer 54 will not continue with the design flow run unless all input data is available. The input data may be stored in individual data files that the user specifies, may be stored in an FCO, or may be stored in RCOs that are referenced in an FCO stored in the flow configuration object storage 36. If the input data is not available, the flow sequencer 54 proceeds to step 174 where it requests the input data from the user. After the data is input, the flow sequencer 54 continues the design flow run in step 176. If the input data is available, the method proceeds directly from step 172 to step 176. The flow sequencer 54 marks the input node inactive and evaluated in step 176. The method then ends.

Referring now to Figure 8, the preferred method of selecting a node to be executed is shown. The selection of the node to be executed is critical since the nodes must be executed in a specific order to enforce design methodologies and other tool input constraints and requirements. Beginning in step 180, the flow sequencer 54 searches the flow map for an active node where all nodes that input to such an active node are inactive. If the flow sequencer 54 finds such a node, the flow sequencer 54 proceeds to step 182. At step 182, the flow sequencer 54 selects the node it found in step 180. The flow sequencer 54 then analyzes the nodes that direct their outputs to the selected node for their status as evaluated. If all the nodes that direct their outputs to the selected node are evaluated, the method ends. If all the nodes that direct their outputs to the selected node are not evaluated, an error has occurred. A node may not be inactive and unevaluated. The system 11 displays an error message through the display device 16 in step 186, and the design flow run is terminated.

If in step 180 there is not an active node having all nodes that direct their outputs to such a node are inactive, the method proceeds to step 188. At step 188, the flow sequencer 54 searches the flow map for active input nodes. If the flow sequencer 54 finds an active input

node in step 188, the flow sequencer 54 proceeds to step 190 where the flow sequencer 54 selects the active input node as the next node to be executed. The flow sequencer 54 then, in step 192, determines whether there is input data with this input node. If there is no input data available, the flow sequencer 54, in step 195, requests input data from the user. When there is input data available, the flow sequencer 54 marks the node inactive and evaluated in step 194 and then returns to step 180 to search for a node that it can execute. The flow sequencer 54 does not select the node into which the input node direct its outputs because there may be other input nodes that direct their outputs to that node. When the flow sequencer 54 returns to step 180, however, it is highly probable that it will be able to select the node into which the input node, which was just evaluated, directs its outputs.

If there are no active input nodes in step 188, the flow sequencer 54 proceeds to step 196 where it determines if there are tool notifications in the tool notification memory 44. If there are no tool notifications, the flow sequencer 54 must wait for a tool notification. Thus, the method returns to step 196. If in step 196 there is a tool notification, the flow sequencer 54, in step 197, selects the tool node, as indicated in the tool notification, that called the design tool. In step 197, the flow sequencer 54 stores the RCO, which is currently stored in the active RCO memory 50, in the run configuration object storage 4U and creates an entry m the FCO, which is currently stored in the active FCO memory 66, of the location of the RCO in the run configuration object storage 40. The flow sequencer 54 creates an entry in the FCO of the active configuration, as stored in the active configuration memory 32. The flow sequencer 54 stores in the FCO in the active FCO memory 66 the version of the RCO, a record of the nodes which produced the input data for the design tool, and a record of the version of the RCO. The flow sequencer 54 then, in step 199, marks the tool node inactive and evaluated. The flow sequencer 54 then returns to step 180 to search for other nodes where all nodes that input to such node are inactive. Once again, it is highly probable that the node to which the tool node that was marked in active in step 199 directs its outputs will be selected in step 182.

Referring now to Figure 9, the preferred method of executing a node is shown. Beginning in step 200, the flow sequencer 54 analyzes the selected node for its status as a tool

node. If the node is a tool node, the flow sequencer 54 proceeds to step 202 where it prepares a new RCO, which it stores in the active RCO memory 50. The flow sequencer 54 then, in step 204, presents the new RCO, the required data, and to the active tool to another system for execution. The flow sequencer 54 presents the RCO, the data, and the active tool to the design tool through the tool launch server 64. Once the flow sequencer 54 has launched the active tool, the method ends.

If in step 200 the node is not a tool node, the flow sequencer 54 proceeds to step 206 where it analyzes the node for its status as a data node. If the node is a data node, the flow sequencer 54 proceeds to step 208 where it gathers the data files specified by the data node. Generally but not necessarily, the data files will have been produced by the design tool executed at the immediately preceding tool node. In this case, the data files will be in, or will be referenced in, the RCO created at the immediately preceding tool node. The data files may, alternatively, be referenced in the active configuration which the flow sequencer 54 recorded in the FCO at the immediately preceding tool node. The flow sequencer 54 then labels, formats, arranges, and otherwise modifies the data, as specified by the data node, so that a design tool at a succeeding tool node can use the data. In step 210, the flow sequencer 54 creates a reference in the FCO of the execution of the data node and marks the data node inactive and evaluated.

If in step 206 the node is not a data node, the flow sequencer 54 proceeds to step 212 where it analyzes the node for its status as a control node. If the node is a control node, the flow sequencer 54 proceeds to step 214 where it executes a function specified by the control node. At a control node, the flow sequencer 54 may analyze any data contained in, or referenced by, the FCO stored in the active FCO memory 66, may analyze the active configuration, may analyze the general design environment, and may analyze any other data available to it. The flow sequencer 54 may, according to the function it executes, change the status of any node from active to inactive or vice versa, or may otherwise modify the design flow run. Thus, at a control node, the flow sequencer 54 may modify the design flow run in light of the current status of the design flow run. The flow sequencer 54 then, in step 216,

creates an entry in the FCO of its actions in response to the control node and marks the control node inactive and evaluated. The method then ends.

If in step 212 the node is not a control node, the flow sequencer 54 proceeds to step 218. At step 218 the flow sequencer 54 determines if the node is a sub-flow node. If the node is a sub-flow node, the flow sequencer 54 proceeds to step 220 where it executes the design flow run specified by the flow map identified by the sub-flow node. The output of the sub-flow node is an FCO that contains a complete record of the design flow run. Note that any design flow run may be executed within a sub-flow node and that design flow runs executed at sub-flow nodes may themselves contain sub-flow nodes. The flow sequencer 54 then, in step 222, makes an entry in the FCO stored in the active FCO memory 66 by recording a reference to the FCO, which was created by executing the sub-flow node. The flow sequencer 54 also marks the sub-flow node inactive and evaluated, and the method ends.

If in step 218 the node is not a sub-flow node, an error has occurred. The method has analyzed the node for all possible nodes and has not found a match. The system 11 outputs an error message through the display device 16 in step 224, and the method ends.

Referring now to Figures 10A and 10B, the preferred method for performing a "flow make" of the system 11 is shown. Flow make is a method of updating a design process after a design flow run has been executed. After a design flow run, design tools, data, or both may be modified. The user needs to identify where the design process has been modified and to update the design process. The system 11 uses flow make to identify the nodes that have been modified and to re-execute those nodes along with all nodes that depend on such modified nodes. Flow make re-executes only the nodes that must be executed to update the design process. Flow make does not execute any node that has not been modified or that does not depend on a node that has been modified.

The method for performing flow make begins in step 260 where the system 11 accepts inputs of a flow map and an FCO. The system 11, in step 262, next creates a working copy of the FCO and stores this copy in the active FCO memory 66. The system 11 creates a working copy of the FCO so that the input FCO is not modified by flow make. The user may, however, override this feature and have the system 11 overwrite the FCO. The system 11

stores the input FCO in the flow configuration object storage 36. The FCO stored in the active FCO memory 66 retains the time stamp of the input FCO. The system 11 then designates the current RCOs in step 264. The current RCOs are those RCOs that users have designated as representing the most current version of a part of the flow map. The system 11 replaces the references to the RCOs in the FCO stored in the active FCO memory 66 with references to the most current RCOs.

In step 266, the flow sequencer 54 marks the relevant nodes of the flow map as active. The user need not execute flow make on the entire flow map. The user can limit flow make to a portion of the flow map. If the user is executing flow make for the entire flow map, all nodes are marked active in step 266. If the user is limiting flow make, then only the relevant nodes are marked active in step 266. The system 11 marks the relevant nodes active using a method that was described with reference to Figures 6A & 6B. The flow sequencer 54 then selects an active input node in step 268. The flow sequencer 54, in step 272, determines whether all nodes that direct their outputs to the current active node are evaluated. If there are unevaluated nodes that input to the current node, the method continues in step 274. In step 274, the flow sequencer 54 searches the flow map in the flow map memory 48 for any active node. If there are no active nodes, the method ends. If there are active nodes, the flow sequencer 54 selects, in step 276, an active node, and the method returns to step 272. If the flow sequencer 54 determines, in step 272, that all nodes that input to the selected node are evaluated, the method continues in step 278.

In step 278, the flow sequencer 54 determines if the node is a tool node. If the node is a tool node, the flow sequencer 54 analyzes the input files, as recorded in the RCO associated with the tool node. The flow sequencer 54 compares the time at which the input files were created to the time recorded in the time stamp of the FCO to determine if all the input files were created before the FCO was completed. If any input files were created after the FCO was completed, the tool node must be re-executed, and the method continues in step 296 where the node is re-executed. If in step 286 all the input files were created before the FCO was completed, the method continues in step 292 where the flow sequencer 54 compares the design tool recorded in the RCO to the design tool specified by the tool node. If the design

tools do not match exactly in step 292, the flow sequencer 54 must re-execute the node. The method continues in step 296 where the flow sequencer 54 executes the node. If in step 292 the design tool specified by the tool node matches the design tool recorded in the RCO, the method continues in step 294. In step 294, the flow sequencer 54 compares the version of the RCO to the version of the FCO. If the versions do not match, the method proceeds to step 296 where the flow sequencer 54 re-executes the node. If in step 294 the version of the RCO matches the version of the FCO, the tool node is current. The flow sequencer 54 marks the tool node inactive and evaluated in step 298, and the method returns to step 276.

If in step 278 the node is not a tool node, the flow sequencer 54 analyzes the node in step 280, to determine if the node is a control node. If the node is a control node, the flow sequencer 54 re-executes the control node in step 296. During flow make, all active control nodes must be re-executed.

If in step 280 the flow sequencer 54 determines that the node is not a control node, the method continues in step 282 where the flow sequencer 54 determines if the node is a data node. If the node is a data node, the flow sequencer 54 analyzes the input data files of the data node as they are recorded in the FCO. The flow sequencer 54 determines if these input data files match the current input data files. If the input data files do not match, the method proceeds to step 296 where the flow sequencer 54 re-executes the data node. If in step 288 the input data files match the input data files recorded in the FCO, the flow sequencer 54 marks the data node inactive and evaluated in step 298, and the method returns to step 276.

Referring now to Figure 11, a flow chart of the preferred method for debugging a design flow run is shown. Similarly to step 100 of Figure 5, a flow map and initial conditions are input to the system 11 in step 310. Included with the flow map and the initial conditions is a list of break points. Break points are nodes at which the user desires to review the data and status of the design flow run. The flow sequencer 54 marks the appropriate nodes active in step 312. The flow sequencer 54 uses the method described above with reference to Figures 6A and 6B. The flow sequencer 54 then initializes a design flow run in step 314; once again, the flow sequencer 54 uses the method described above with reference to Figure 7. In step 316, the flow sequencer 54 selects an active node. After the flow sequencer 54 selects a node

in step 316, the flow sequencer 54 preferably searches the list of break points, in step 318, to determine if the selected node is a break point. If the selected node is not a break point, the method continues to step 332. If at step 318 the selected node is determined to be a break point, the flow sequencer 54, in step 320 determines if choice 1, choice 2, or choice 3 is associated with this break point.

The flow sequencer 54 may take one of three actions at a break point. The choice of action is a part of the initial conditions and is associated with the break points. In choice 1, the flow sequencer 54 outputs, through the display device 16, the input files to the node that is the break point and then pauses. The flow sequencer 54 searches for a command, which is input through the input device 10, to continue. When the command to continue is input, the flow sequencer 54 continues the design flow run. In choice 2, the flow sequencer stores the FCO, which is currently stored in the active FCO memory 66, in the flow configuration object storage 36. The flow sequencer 54 then continues the design flow run. In choice 3, the flow sequencer 54 outputs the data, which is input to the selected node, stores the FCO, which is currently stored in the active FCO memory 66, in the flow configuration storage 36, and ends the design flow run. Choices 1, 2, and 3 may be used in any combination with the break points. For example, the first break point may be associated with choice 2, and the second break point may be associated with choice 1. Since choice 3 ends the design flow run, only one break point associated with choice 3 can be executed during a design flow run.

If in step 318 the flow sequencer 54 determines that the selected node is a break point, the flow sequencer 54, in step 320, determines if choice 1, choice 2, or choice 3 is associated with the break point. If choice 1 is associated with the break point, the flow sequencer 54, in step 322, outputs the data, which is the input data to the selected node, through the display device 16. The flow sequencer 54 then, step 324, searches for a command to continue. When the flow sequencer 54 finds the command to continue, the method proceeds to step 332. If in step 320 the flow sequencer 54 determines that choice 2 is associated with the break point, the flow sequencer 54 in step 326, stores the FCO, which is currently stored in the active FCO memory 66, in the flow configuration storage 36. The method then continues in step 332. If in step 320 the flow sequencer 54 determines that choice 3 is associated with the break point,

the flow sequencer 54 stores the FCO, which is currently stored in the active FCO memory 66, in the flow configuration object storage 36. The flow sequencer 54 then, in step 330, outputs the data, which is the input data to the node, through the display device 16. The method then ends.

If in step 318 the flow sequencer 54 determines that the node is not a break point, the method continues in step 332. In step 332, the flow sequencer 54 executes the node. The flow sequencer 54 uses the method discussed above with reference to Figures 8 and 9. The flow sequencer 54 then, in step 334, searches the flow map stored in the flow map memory 48 for active nodes. If the flow sequencer 54 finds an active node, the method returns to step 316. If the flow sequencer 54 does not find, in step 334, an active node the method ends.

While the present invention has been described with reference to certain preferred embodiments, those skilled in the art will recognize that various modifications may be provided. For example, the data contained in the RCOs may be stored directly in the active FCO or the design tools may be executed either by the processor or by separate parallel processors. These and other variations upon and modifications to the preferred embodiments are provided for by the present invention, which is limited only by the following claims.

WHAT IS CLAIMED IS:

1. An apparatus for enforcing design methodologies, analyzing design processes, and creating histories of design processes comprising:

a processor, having inputs and outputs, for executing program instruction steps;

a flow map memory, having inputs and outputs coupled to the inputs and outputs of the memory, for storing a flow map in response to commands from the processor;

an active FCO memory, having inputs and outputs coupled to the inputs and outputs of the processor, for storing a flow configuration object in response to commands from the processor; and

a flow sequencer memory, having inputs and outputs coupled to the inputs and outputs of the processor, the flow sequencer memory storing program instruction steps for controlling the processor for executing the flow map stored in the flow map memory and for constructing the flow configuration object stored in the active FCO memory.

2. The apparatus of claim 1, wherein the memory further comprises an active RCO memory, having inputs and outputs coupled to the inputs and outputs of the processor, for storing a run configuration object in response to commands from the processor.

3. The apparatus of claim 2 further comprising:

an input device, having outputs coupled to the processor, for the inputting of data and commands;

a display device, having inputs coupled to the processor, for outputting data in response to commands from the processor; and

a data storage device, having inputs and outputs coupled to the processor, and to the display device, for storing a flow configuration object, a run configuration object, and a flow map in response to commands from the processor.

4. The apparatus of claim 3, wherein the data storage device further comprises;

a flow configuration object storage, having inputs and outputs coupled to the processor, for storing flow configuration objects in response to commands from the processor;

a flow map storage, having inputs and outputs coupled to the processor, for storing flow maps in response to commands from the processor;

a design tool storage, having inputs and outputs coupled to the processor, for storing design tools in response to commands from the processor; and

a run configuration object storage, having inputs and outputs coupled to the processor, for storing run configuration objects in response to commands from the processor.

5. The apparatus of claim 4, wherein the memory further comprises:

a flow file description memory, havirig inputs and outputs coupled to the inputs and outputs of the processor, for storing a flow file description in response to commands from the processor;

a flow file reader memory, having inputs and outputs the inputs coupled to receive a flow file description from the flow file description memory and coupled to the inputs and outputs of the processor, for storing program instruction steps for controlling the processor for reading and parsing the flow file description stored in the flow file description memory; and

a flow generator memory, having inputs and outputs coupled to the inputs and outputs of the processor and coupled to receive data from the flow file reader memory, for storing program instruction steps for controlling the processor for constructing a flow map from the output of the flow file reader and for storing the flow map in the flow map memory.

6. The apparatus of claim 5, wherein the memory further comprises a flow editor memory, having inputs and outputs coupled to the inputs and outputs of the processor, for storing program instruction steps for controlling the processor for permanently modifying the flow file description stored in the flow file description memory, the flow map stored in the

flow map memory the flow sequencer, and the flow configuration object stored in the active FCO memory in response to commands from the input device and the processor.

7. The apparatus of claim 6, wherein the memory further comprises a flow viewer, having inputs and outputs coupled to the inputs and outputs of the processor, for temporarily modifying the flow map stored in the flow map memory, the flow sequencer, and the flow configuration object stored in the active FCO memory in response to commands from the input device and the processor.

8. The apparatus of claim 7, wherein the memory further comprises:

an active configuration memory, having inputs and outputs coupled to the inputs and outputs of the processor, for storing a record of the current design environment and default conditions of the apparatus in response to commands from the processor;

an active tool memory, having inputs and outputs coupled to the inputs and outputs of the processor, for storing a record of the design tool currently being executed in response to commands from the processor;

a tool notification memory, having inputs and outputs coupled to the inputs and outputs of the processor, for storing a signal to the flow sequencer that a design tool has completed execution and the node from which the design tool was executed in response to commands from the processor;

a graphical user interface memory, having inputs and outputs coupled to the inputs and outputs of the processor, for storing at least one graphical user interface in response to commands from the processor; and

a tool launch server, having inputs and outputs coupled to the inputs and outputs of the processor, for storing program instruction steps for controlling the processor for creating a conduit to systems for the execution of design tools.

9. A method for creating design configurations and for controlling the execution of multiple design tools, the method comprising the steps of:

preparing nodes of a flow map that are to be executed;

determining if there are nodes of the flow map to be executed;

selecting a node of the flow map;

executing the selected node; and

repeating the steps of selecting and executing for each node of the flow map to be executed and that is unexecuted.

10. The method of claim 9 wherein the method of preparing the nodes of the flow map comprises the steps of:

determining if all nodes are to be executed;

marking the selected nodes active if not all nodes are to be executed;

marking all nodes active if all nodes are to be executed; and

marking all active nodes unevaluated.

11. The method of claim 10 wherein the method of marking the selected nodes of the flow map active comprises the steps of:

receiving a node as a node as a node of interest;

determining if the node of interest is an input node;

marking nodes, which depend upon the node of interest, active if the node of interest is an input node; and

marking nodes, upon which the node of interest depends, active if the node of interest is not an input node.

12. The method of claim 10 wherein the method of selecting an active node comprises the steps of:

selecting an active node where all nodes that direct their outputs to the selected node are inactive and evaluated;

selecting an active input node if there are no nodes where all nodes that direct their outputs to the node are inactive and evaluated;

marking the node inactive and evaluated if an input node is selected;

repeating this method if an input node was selected;

determining if there is a tool notification;

repeating the step of determining if there is no tool notification until there is a tool notification;

selecting the node indicated in the tool notification;

marking the node, indicated in the tool notification, inactive and evaluated; and

repeating this method if there was a tool notification.

13. The method of claim 9 wherein the method of executing an active node comprises the steps of:

determining if the node is a tool node;

preparing a run configuration object and storing the run configuration object in an active RCO memory if the node is a tool node;

executing the design tool if the node is a tool node;

determining if the node is a data node;

preparing data in accordance with the data node if the node is a data node;

determining if the node is a control node;

executing a function defined by the control node if the node is a control node;

determining if the node is a sub-flow node; and

executing a design flow run specified by the sub-flow node if the node is a sub-flow node.

14. A method for updating a design process comprising the steps to: selecting a node;

determining if all inputs to the node are current;

determining if the node is current; and

executing the node if either the inputs to the node or if the node are not current.

15. A method for debugging a design process comprising the steps of:

inputting a flow map and a list of break points;

selecting an active node of the flow map;

determining if the selected node is a break point;

outputting the inputs to the selected node if the selected node is a break point;

executing the node; and

repeating the steps of selecting, determining, outputting, executing, and repeating.

16. The method of claim 15 further comprising the step of storing an flow configuration object which is being constructed, in a flow configuration storage if a selected node is a break node.

17. A system for creating design configurations and for controlling the execution of multiple design tool comprising:

a first storage means for storing a flow map, having nodes, for execution;

a flow sequencer means, coupled to the first storage means, for determining the nodes of the flow map to be executed, for executing the nodes of the flow map, and for constructing a history of the design process; and

a second storage means, coupled to the flow sequencer means, for storing the history of the design process.

18. The system of claim 17 further comprising a third storage means, having inputs and outputs coupled to the flow sequencer means and the second storage means, for storing a history of a design tool run referenced in the history of the design process stored in the second storage means.

19. The apparatus of claim 18 further comprising:

a flow file description memory means, having inputs and outputs, for storing a description of flow map;

an input means, having inputs and outputs coupled to the flow file description memory means, for inputting data and commands;

a flow file reader means, having inputs and outputs the inputs coupled to the flow file description memory means, for reading and parsing the data stored in the flow file description means; and

a flow generator means, having inputs coupled to the flow file reader means and outputs coupled to the first storage means, for generating a flow map from the outputs of the flow file description memory means and for storing the generated flow map in the first storage means.

20. The apparatus of claim 19 further comprising an editor means, coupled to the flow file description memory means, the flow sequencer means, the first storage means, the

second storage means, and the third storage means, for modifying permanently the first storage means, the flow sequencer means, the second storage means, the third storage means and the flow file description memory means.

21. The apparatus of claim 20 further comprising a viewer means, having inputs and outputs coupled to the flow sequencer means, the first storage means, the second storage means, and the third storage means, for modifying temporarily the first storage means, the flow sequencer means, the second storage means, and the third storage means.

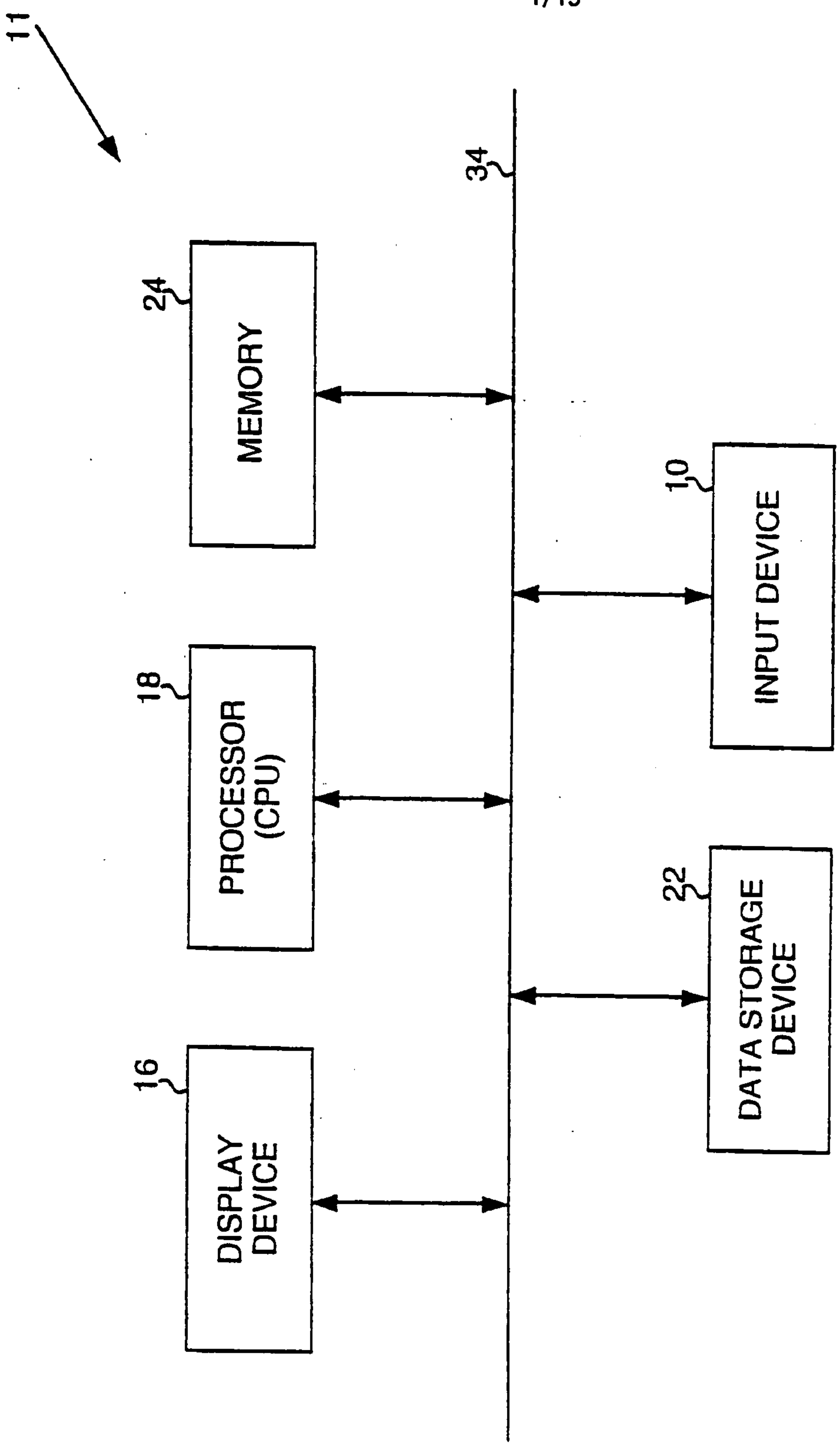

11
34
24
MEMORY
18
PROCESSOR (CPU)
16
DISPLAY DEVICE
10
INPUT DEVICE
22
DATA STORAGE DEVICE

FIG. 1

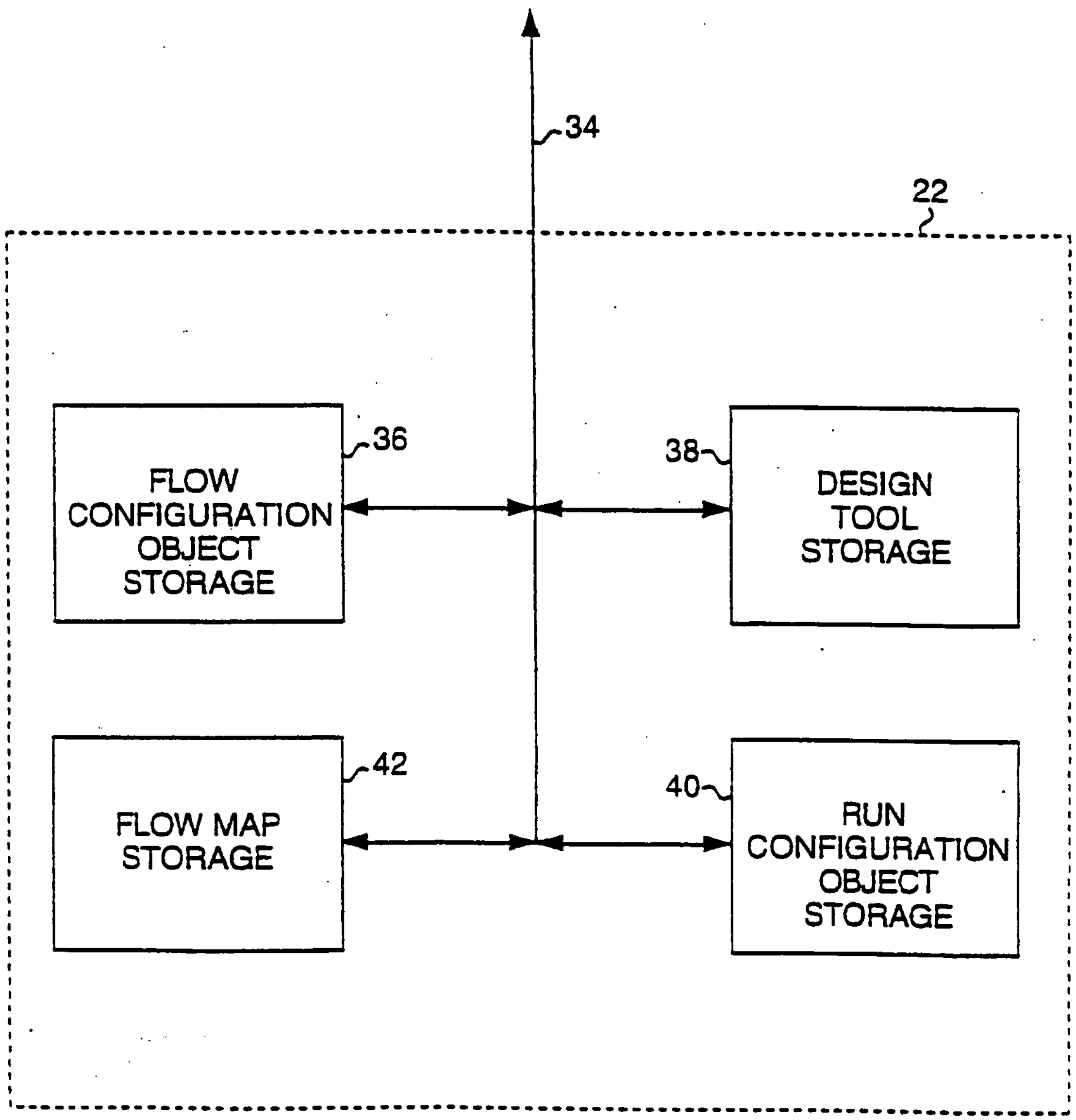
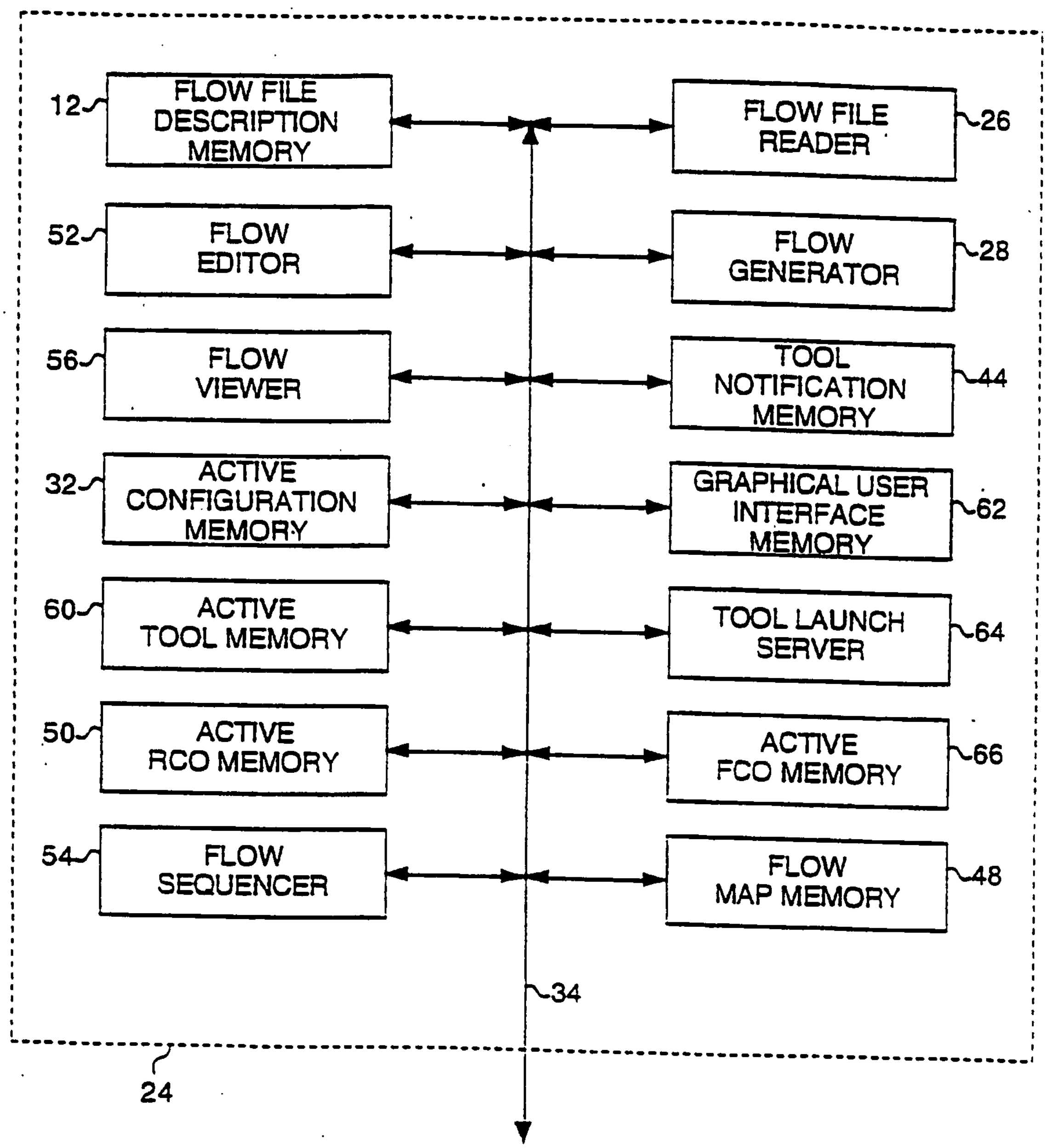
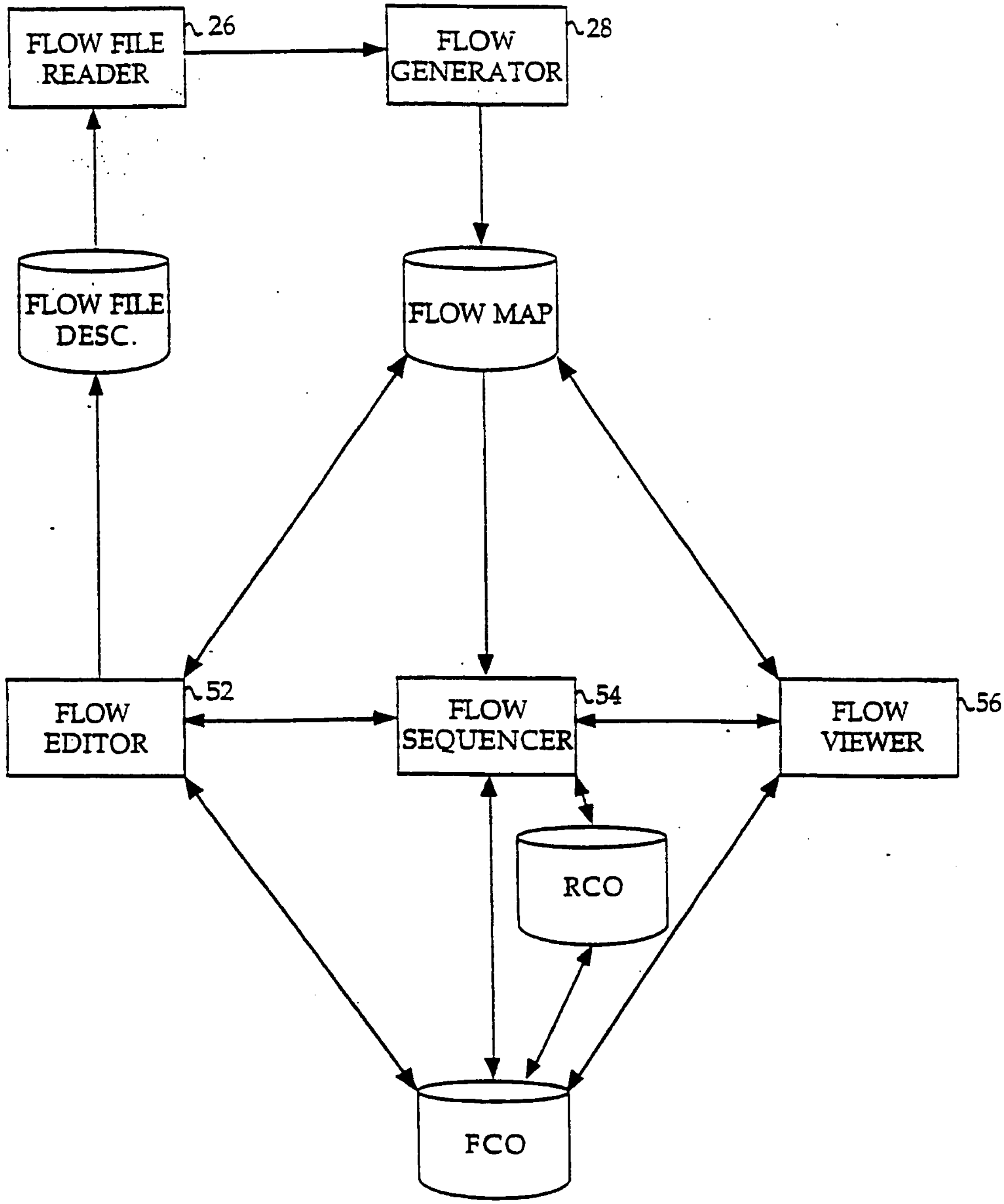

34
22
FLOW CONFIGURATION OBJECT STORAGE
36
38
DESIGN TOOL STORAGE
FLOW MAP STORAGE
42
40
RUN CONFIGURATION OBJECT STORAGE

FIG. 2



FLOW FILE DESCRIPTION MEMORY
12
FLOW EDITOR
52
FLOW VIEWER
56
ACTIVE CONFIGURATION MEMORY
32
ACTIVE TOOL MEMORY
60
ACTIVE RCO MEMORY
50
FLOW SEQUENCER
54
FLOW FILE READER
26
FLOW GENERATOR
28
TOOL NOTIFICATION MEMORY
44
GRAPHICAL USER INTERFACE MEMORY
62
TOOL LAUNCH SERVER
64
ACTIVE FCO MEMORY
66
FLOW MAP MEMORY
48
34
24

FIG. 3



FLOW FILE READER
26
FLOW GENERATOR
28
FLOW FILE DESC.
FLOW MAP
FLOW EDITOR
52
FLOW SEQUENCER
54
FLOW VIEWER
56
RCO
FCO

FIG. 4

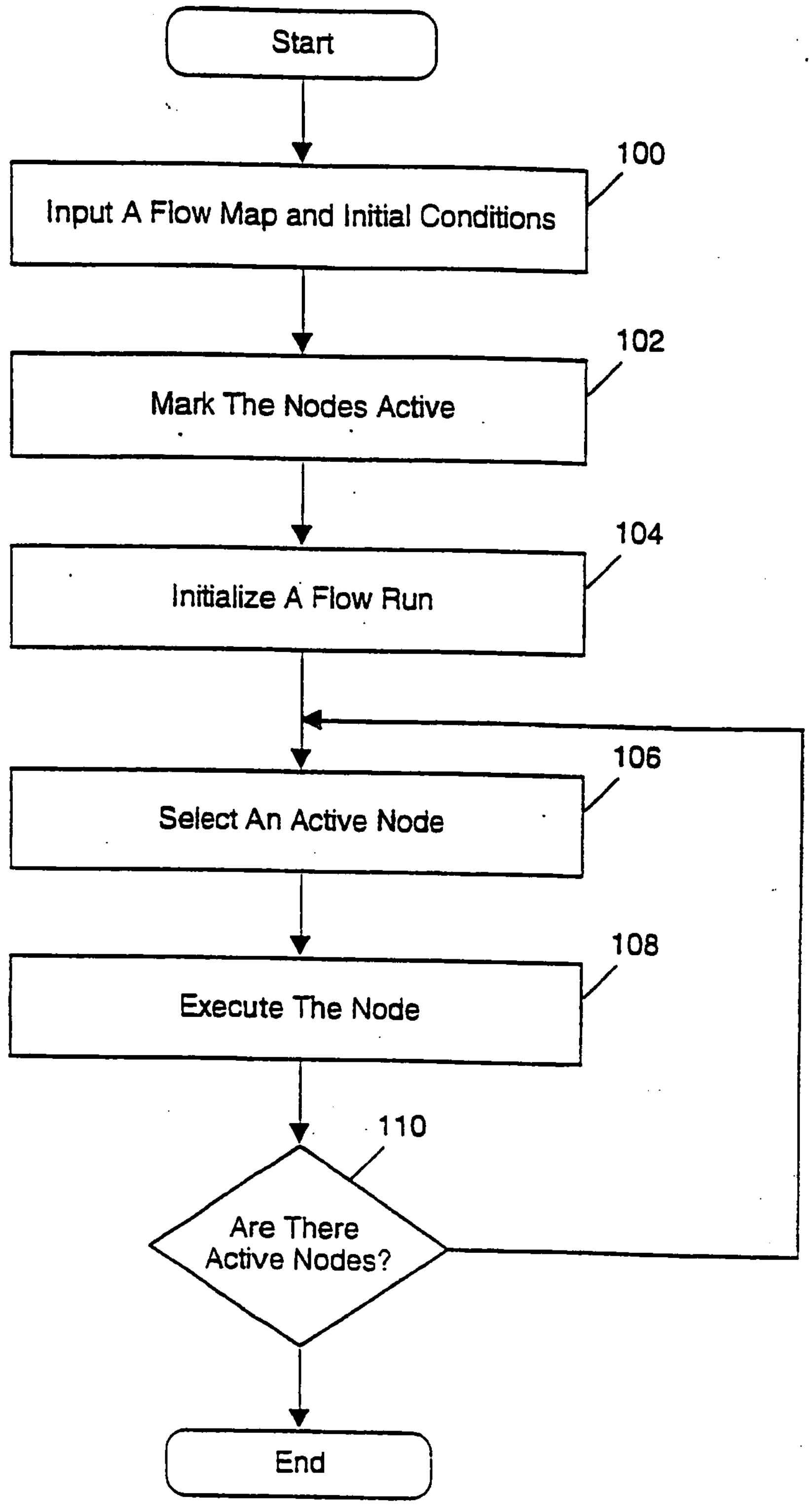
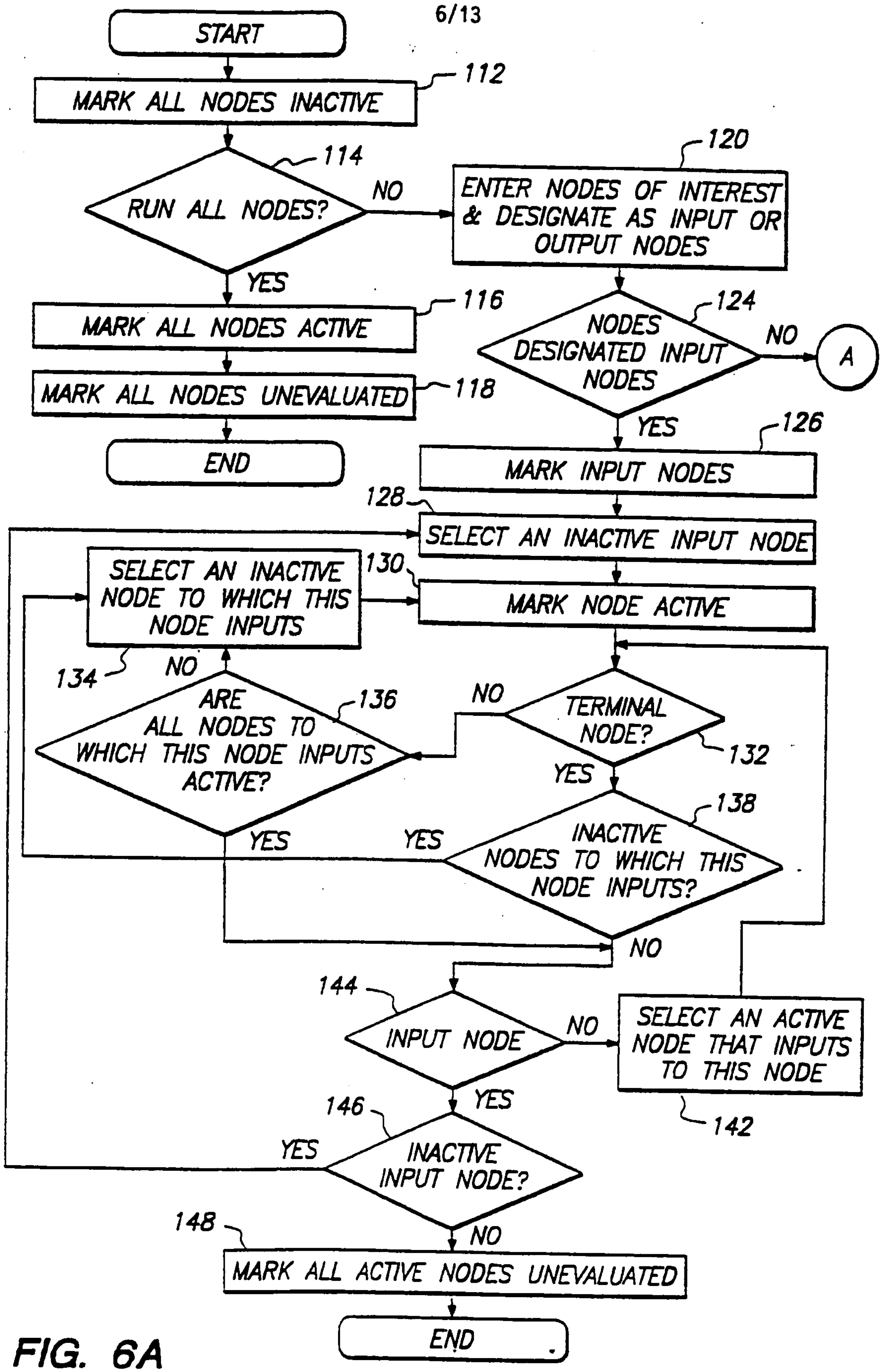

Start
Input A Flow Map and Initial Conditions
100
Mark The Nodes Active
102
Initialize A Flow Run
104
Select An Active Node
106
Execute The Node
108
110
Are There Active Nodes?
End

FIG. 5



START
MARK ALL NODES INACTIVE
112
RUN ALL NODES?
114
NO
YES
ENTER NODES OF INTEREST & DESIGNATE AS INPUT OR OUTPUT NODES
120
MARK ALL NODES ACTIVE
116
MARK ALL NODES UNEVALUATED
118
END
NODES DESIGNATED INPUT NODES
124
NO
A
YES
MARK INPUT NODES
126
128
SELECT AN INACTIVE INPUT NODE
SELECT AN INACTIVE NODE TO WHICH THIS NODE INPUTS
130
MARK NODE ACTIVE
134
NO
ARE ALL NODES TO WHICH THIS NODE INPUTS ACTIVE?
136
YES
NO
TERMINAL NODE?
132
YES
INACTIVE NODES TO WHICH THIS NODE INPUTS?
138
YES
NO
144
INPUT NODE
NO
SELECT AN ACTIVE NODE THAT INPUTS TO THIS NODE
142
YES
146
INACTIVE INPUT NODE?
YES
NO
148
MARK ALL ACTIVE NODES UNEVALUATED
END

FIG. 6A

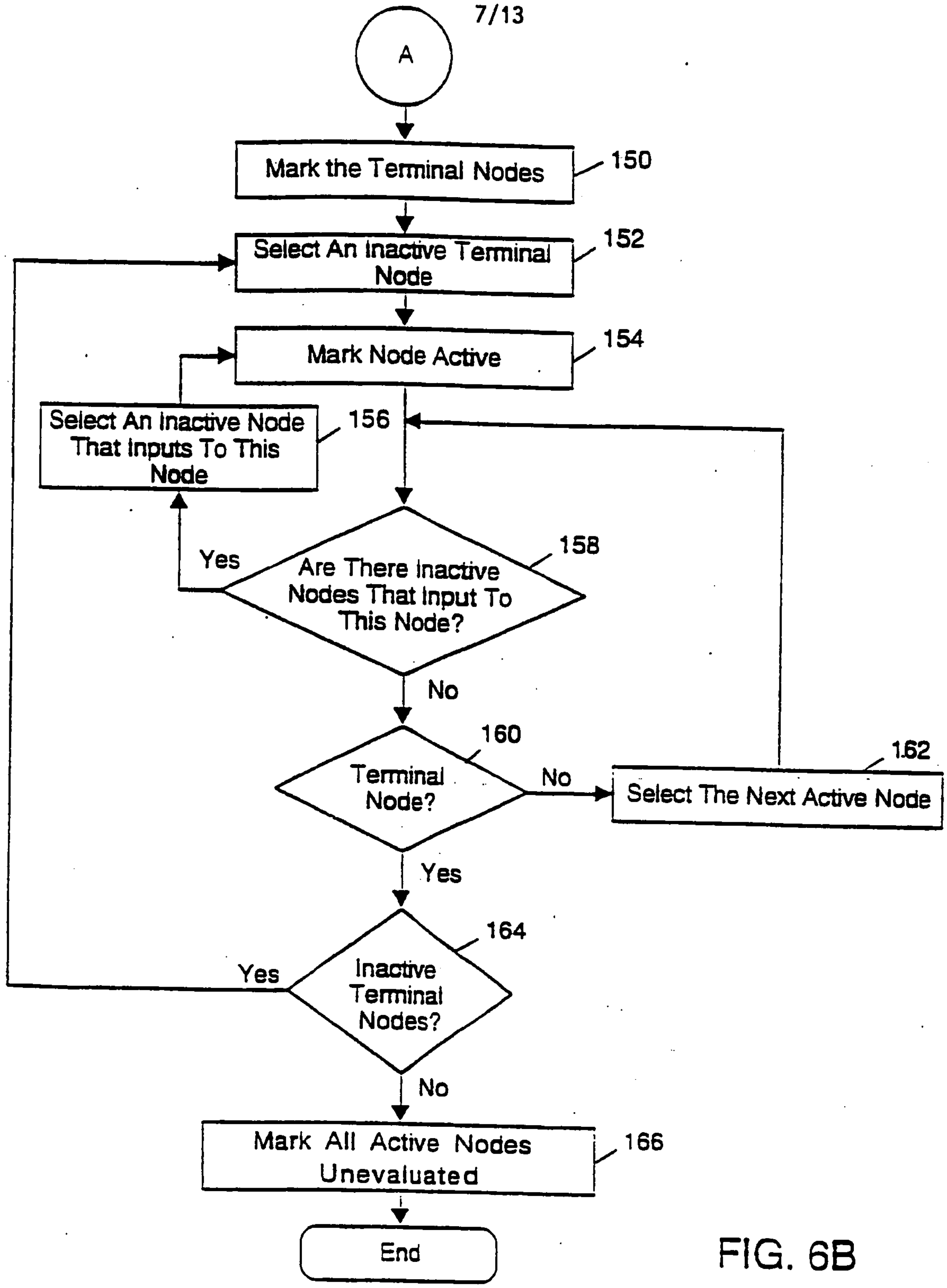

A
Mark the Terminal Nodes
150
Select An Inactive Terminal Node
152
Mark Node Active
154
Select An Inactive Node That Inputs To This Node
156
Yes
Are There Inactive Nodes That Input To This Node?
158
No
Terminal Node?
160
No
Select The Next Active Node
162
Yes
Inactive Terminal Nodes?
164
Yes
No
Mark All Active Nodes Unevaluated
166
End

FIG. 6B

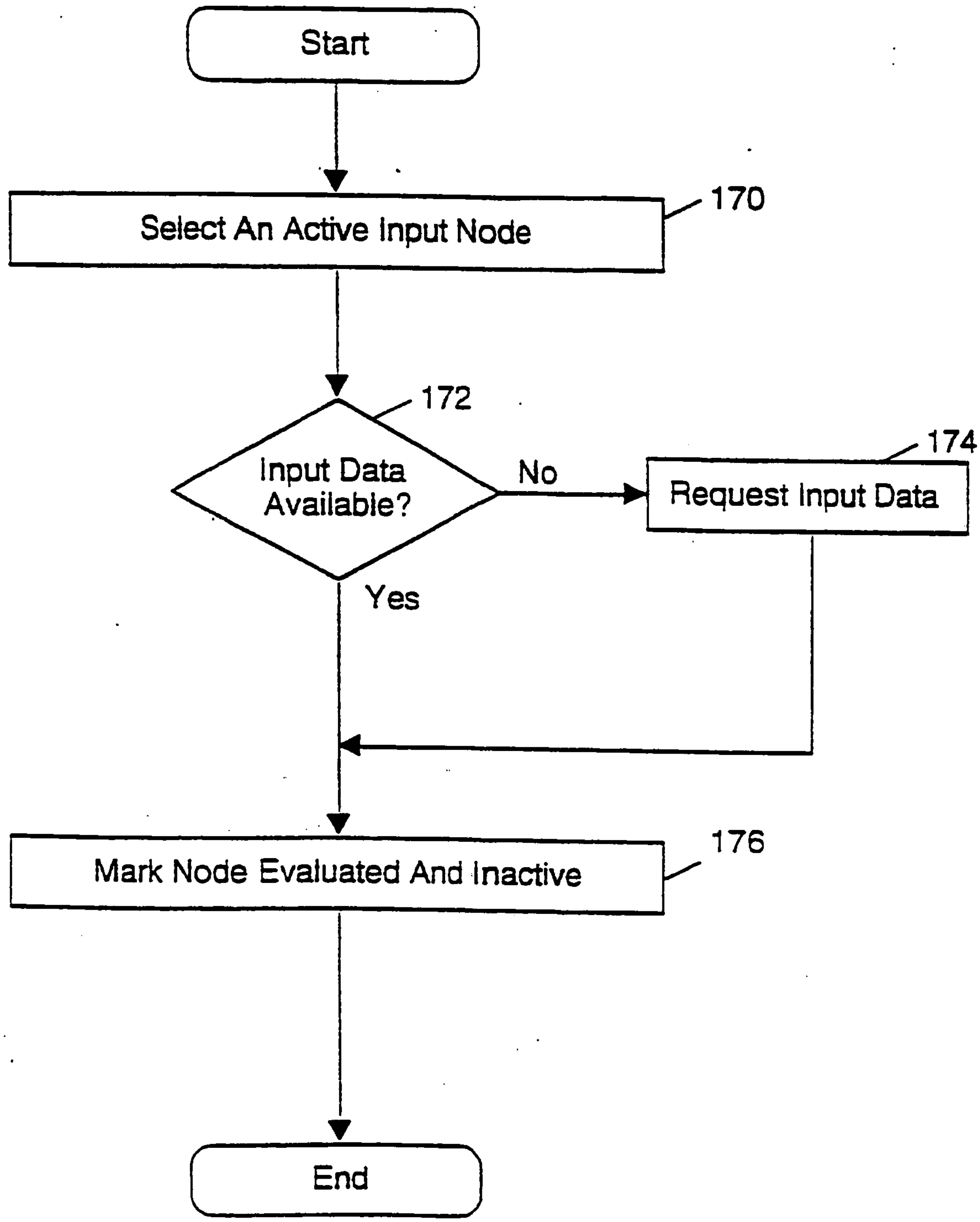

Start
Select An Active Input Node
170
172
Input Data Available?
No
174
Request Input Data
Yes
176
Mark Node Evaluated And Inactive
End

FIG. 7

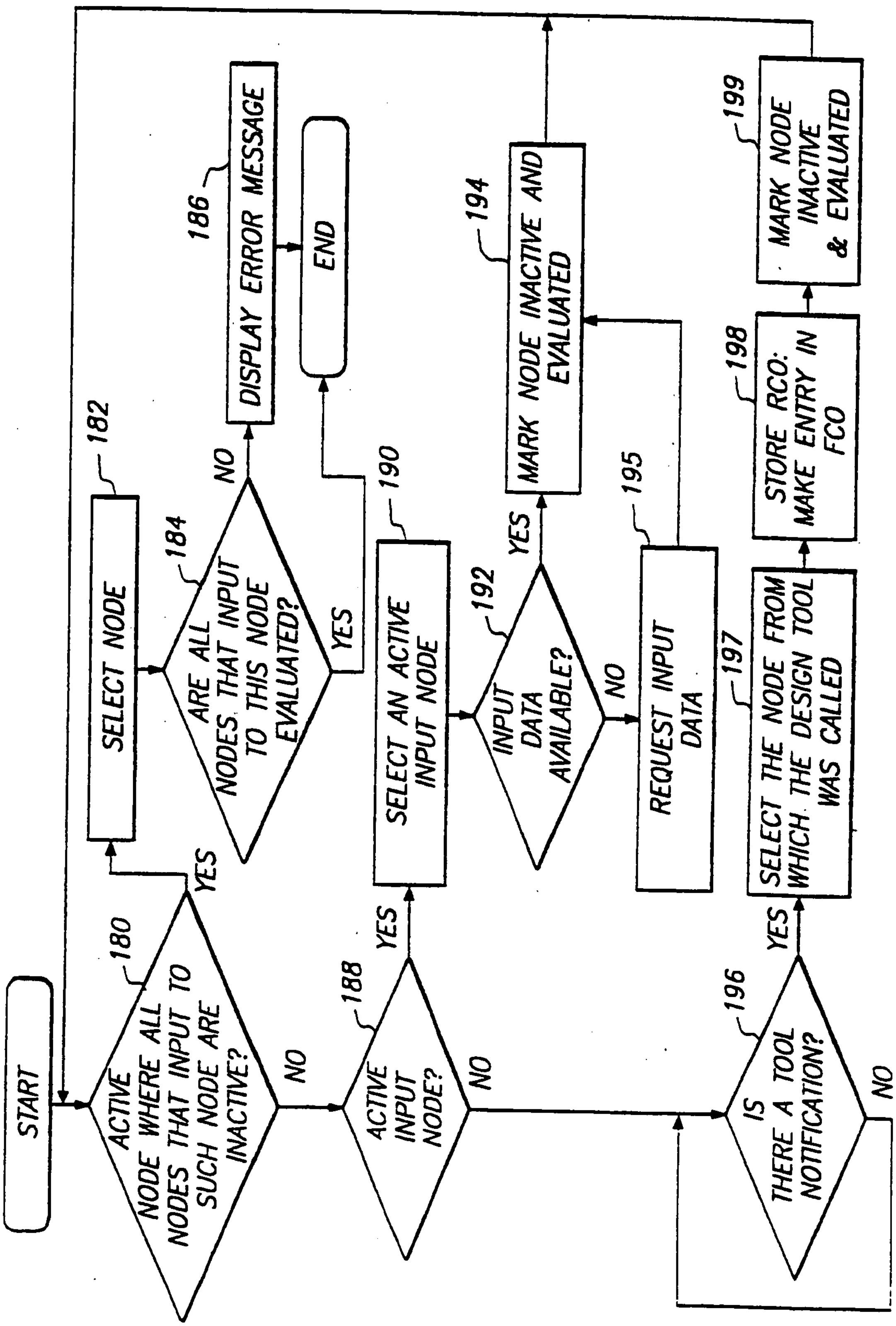

START
ACTIVE NODE WHERE ALL NODES THAT INPUT TO SUCH NODE ARE INACTIVE?
180
YES
NO
SELECT NODE
182
ARE ALL NODES THAT INPUT TO THIS NODE EVALUATED?
184
YES
NO
DISPLAY ERROR MESSAGE
186
END
ACTIVE INPUT NODE?
188
YES
NO
SELECT AN ACTIVE INPUT NODE
190
INPUT DATA AVAILABLE?
192
YES
NO
MARK NODE INACTIVE AND EVALUATED
194
REQUEST INPUT DATA
195
IS THERE A TOOL NOTIFICATION?
196
YES
NO
SELECT THE NODE FROM WHICH THE DESIGN TOOL WAS CALLED
197
STORE RCO: MAKE ENTRY IN FCO
198
MARK NODE INACTIVE & EVALUATED
199

FIG. 8

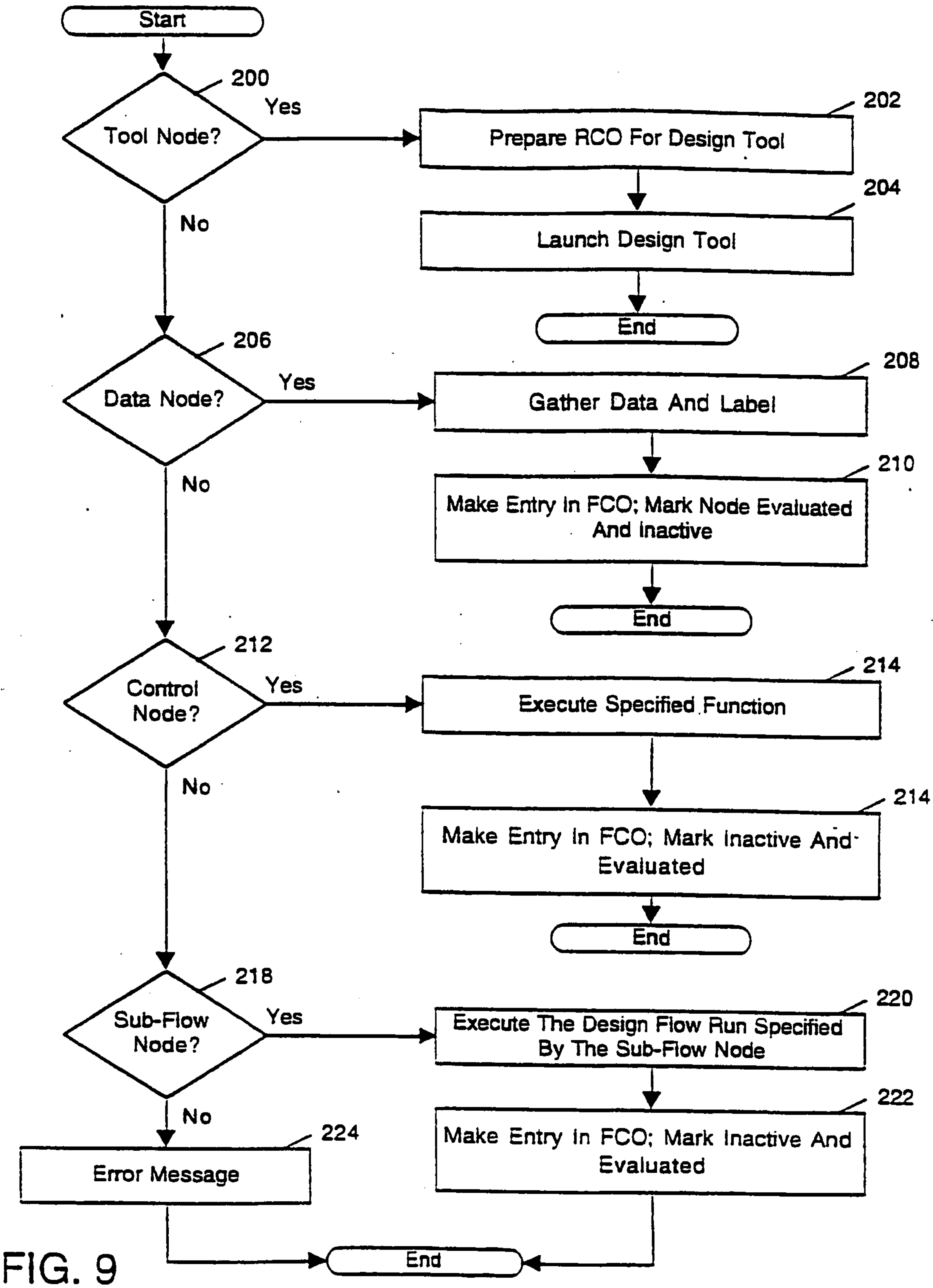

Start
200
Tool Node?
Yes
202
Prepare RCO For Design Tool
204
Launch Design Tool
End
No
206
Data Node?
Yes
208
Gather Data And Label
210
Make Entry In FCO; Mark Node Evaluated And Inactive
End
No
212
Control Node?
Yes
214
Execute Specified Function
214
Make Entry In FCO; Mark Inactive And Evaluated
End
No
218
Sub-Flow Node?
Yes
220
Execute The Design Flow Run Specified By The Sub-Flow Node
222
Make Entry In FCO; Mark Inactive And Evaluated
No
224
Error Message
End

FIG. 9

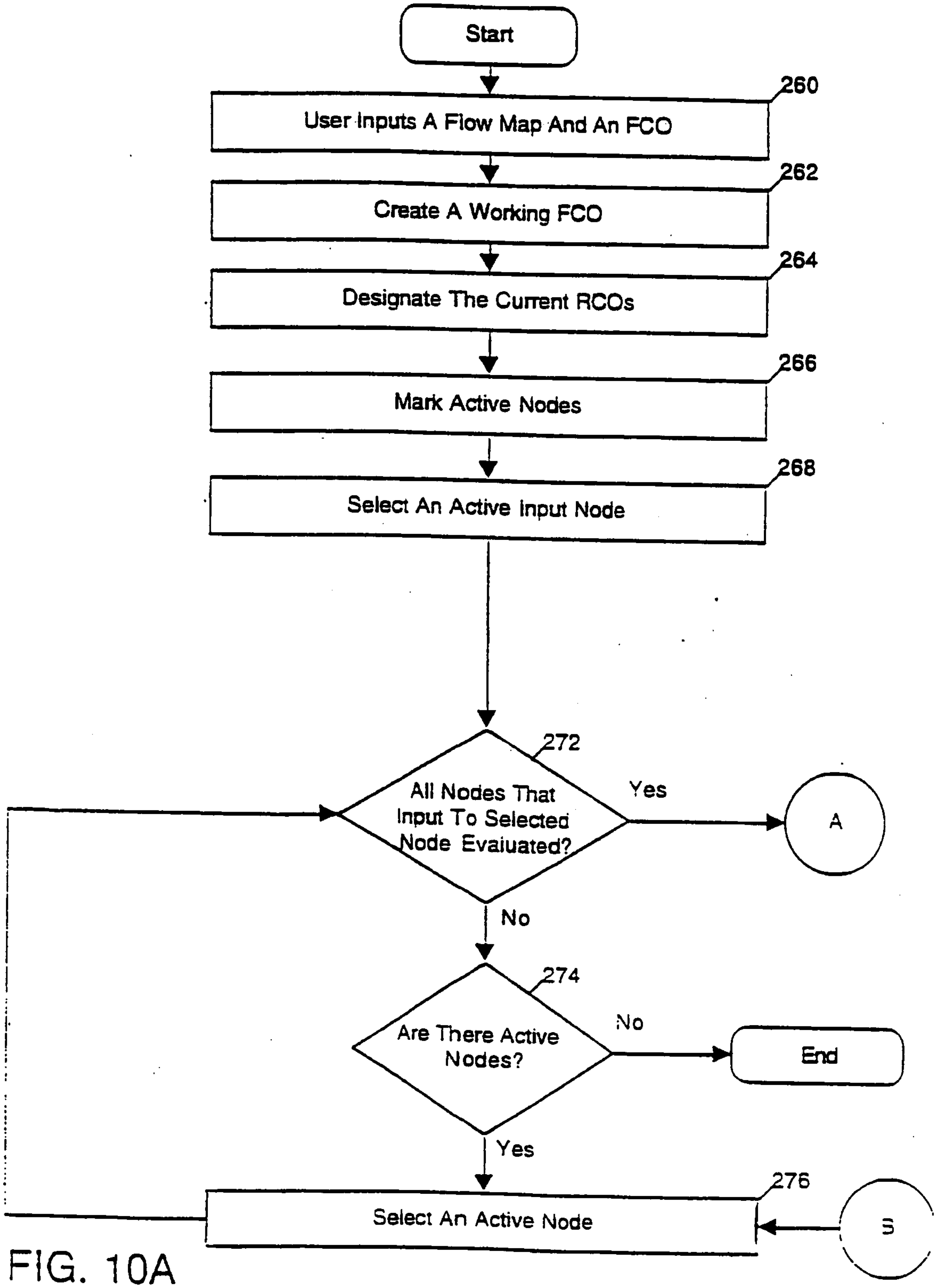

Start
260
User Inputs A Flow Map And An FCO
262
Create A Working FCO
264
Designate The Current RCOs
266
Mark Active Nodes
268
Select An Active Input Node
272
All Nodes That Input To Selected Node Evaluated?
Yes
A
No
274
Are There Active Nodes?
No
End
Yes
276
Select An Active Node
B

FIG. 10A

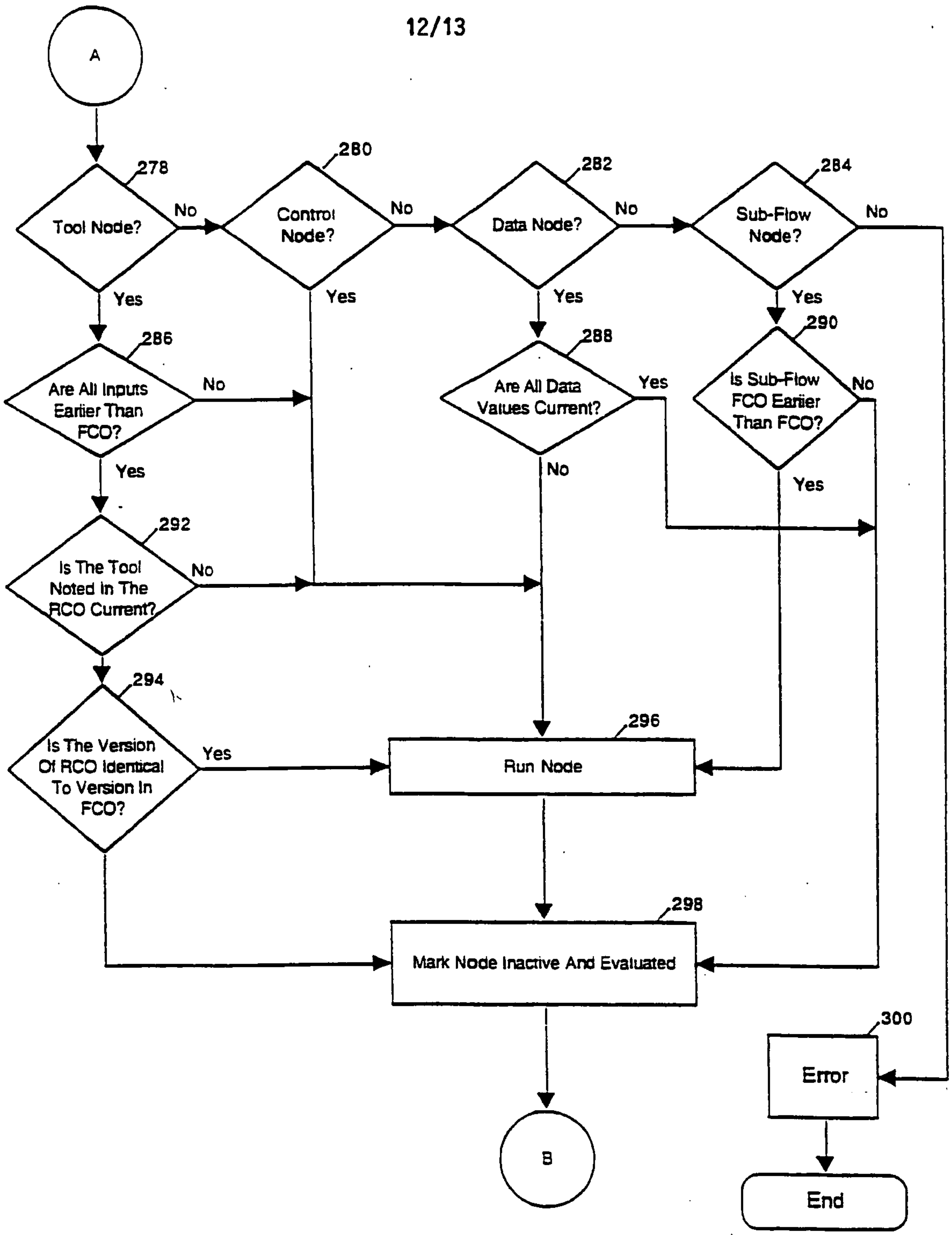

A
Tool Node?
278
No
Yes
Control Node?
280
No
Yes
Data Node?
282
No
Yes
Sub-Flow Node?
284
No
Yes
Are All Inputs Earlier Than FCO?
286
No
Yes
Are All Data Values Current?
288
Yes
No
Is Sub-Flow FCO Earlier Than FCO?
290
No
Yes
Is The Tool Noted In The RCO Current?
292
No
Is The Version Of RCO Identical To Version In FCO?
294
Yes
Run Node
296
Mark Node Inactive And Evaluated
298
Error
300
End
B

FIG. 10B

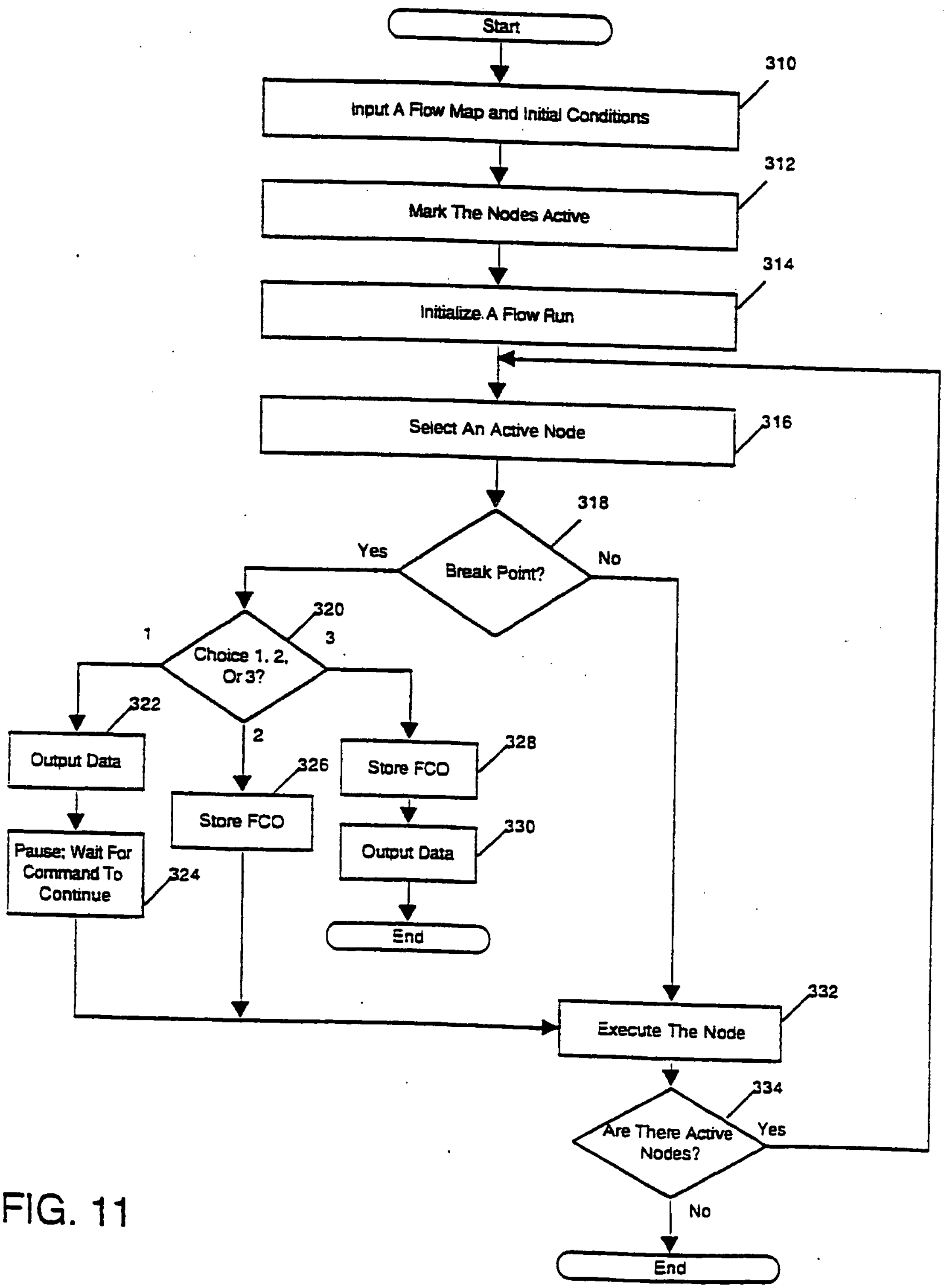

Start
Input A Flow Map and Initial Conditions
310
Mark The Nodes Active
312
Initialize A Flow Run
314
Select An Active Node
316
Break Point?
318
Yes
No
Choice 1, 2, Or 3?
320
1
2
3
Output Data
322
Pause; Wait For Command To Continue
324
Store FCO
326
Store FCO
328
Output Data
330
End
Execute The Node
332
Are There Active Nodes?
334
Yes
No
End

FIG. 11

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 95/06791

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F15/60 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IEEE INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 11 November 1990, SANTA CLARA CA US pages 482 - 485 VAN DEN HARMER ET AL 'a data flow based architecture for cad frameworks' see page 482, column 2, line 19 - page 483, column 1, line 9; figures 1-3 ---	1-21
A	IEEE PROCEEDINGS OF THE 30TH DESIGN AUTOMATION CONFERENCE, 14 June 1993, DALLAS US pages 648 - 653 SUTTON ET AL 'design management using dynamically defined flows' see page 650, column 1, line 10 - column 2, line 26; figures 3-5 ---	1-21
	-/--	

[X] Further documents are listed in the continuation of box C.

[ ] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
24 August 1995	18.09.95

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Guingale, A

Form PCT/ISA/210 (second sheet) (July 1992)

International Application No
PCT/US 95/06791

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IEEE TRANSACTIONS ON COMPUTER AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, vol.12, no.8, August 1993, NEW YORK US pages 1077 - 1095 CASOTTO ET AL 'automated design management using traces' see page 1080, column 1, line 54 - page 1083, column 1, line 3; figure 1 ---	1-21
A	IEEE INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 8 November 1992, SANTA CLARA US pages 538 - 545 BINGLEY ET AL 'incorporating design flow management in a framework based cad system' see the whole document -----	1-21

1